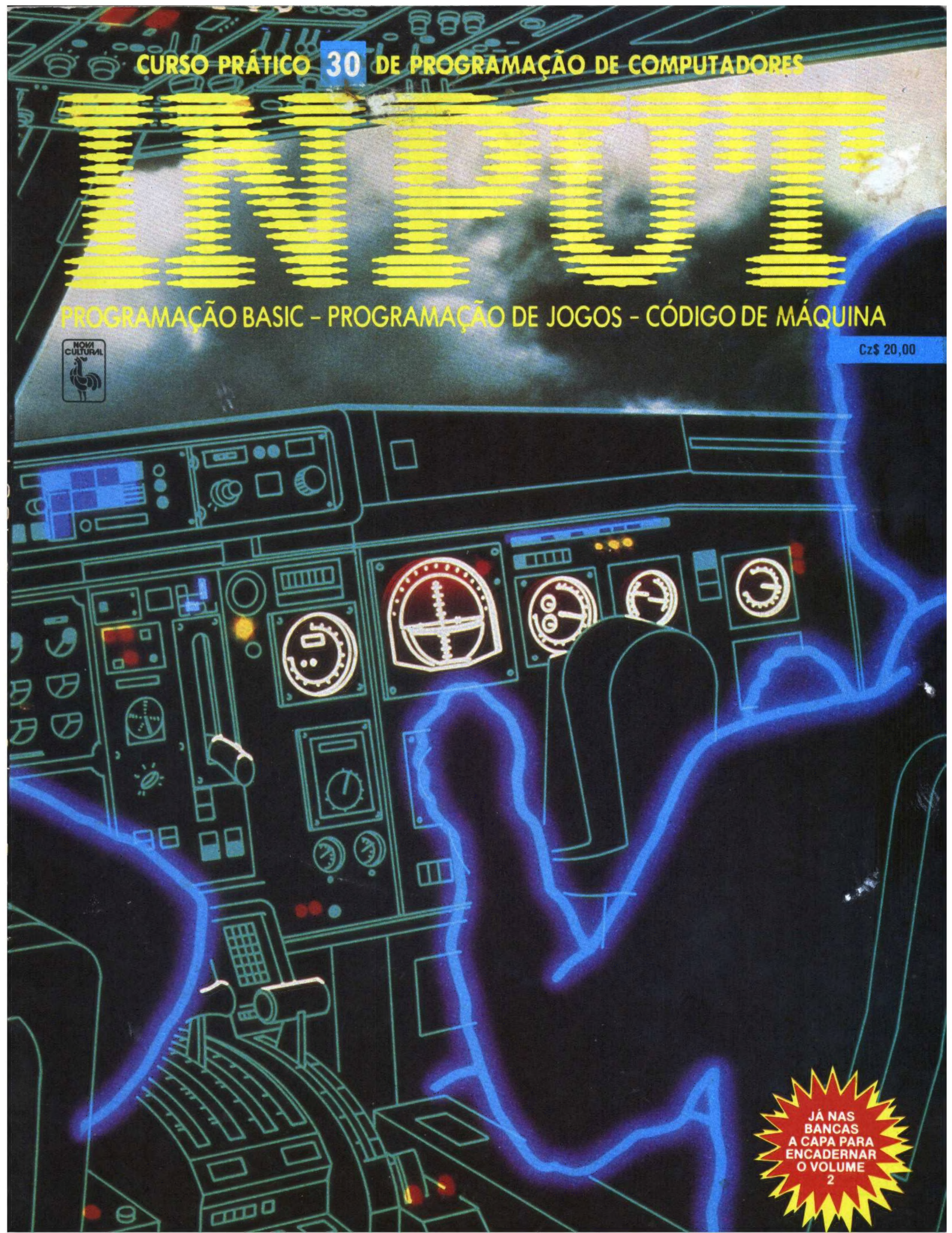
CURSO PRÁTICO 30 DE PROGRAMAÇÃO DE COMPUTADORES

INFO

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 20,00



JÁ NAS BANCAS A CAPA PARA ENCADERNAR O VOLUME 2

INPUT

Vol. 2 Nº 30

NESTE NÚMERO

PROGRAMAÇÃO BASIC

PROGRAMAÇÃO DE GRÁFICOS EM 3-D (1)

Desenhos tipo *wireframe*. Organização e incorporação das rotinas. Inicialização da máquina. Desenho de uma grade. Desenho de círculo..... 581

APLICAÇÕES

UM EDITOR DE TEXTOS (2)

Correção de erros ortográficos. Composição. Armazenagem. Edição em cada máquina..... 586

PROGRAMAÇÃO DE JOGOS

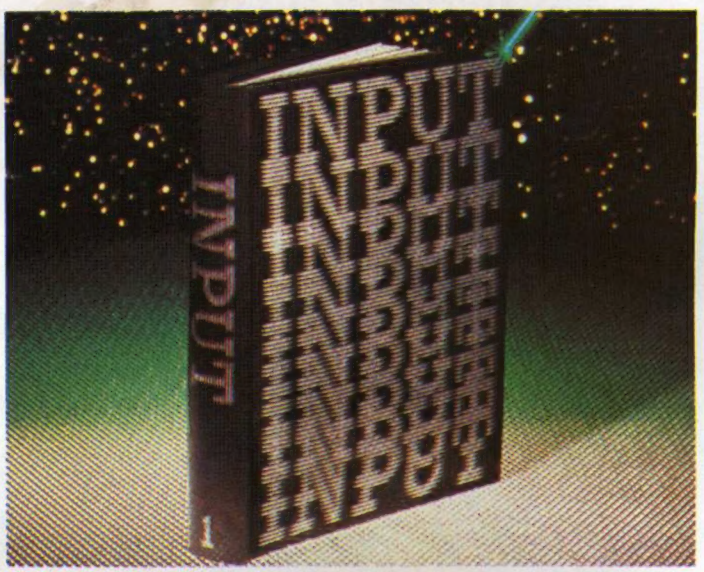
UM SIMULADOR DE VÔO (1)

O que é simulação de vôo. Programas de treinamento. Vôo por instrumentos. O painel do avião. Como criar a sensação de movimento..... 592

CÓDIGO DE MÁQUINA

AMPLIE O BASIC DO TRS-COLOR

Planeje novas instruções. Adição de *stubs*. Reconhecimento de novos comandos. Recuperação de programas. Troca do conjunto de cores 597



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: **1. Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). **2. Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o nº (011) 33670 ABSA. Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.
Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção**: Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretora Editorial: Iara Rodrigues

Editor Executivo: Antonio José Filho

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, Grace Alonso Arruda, José Maria de Oliveira, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução: Reinaldo Cúrcio
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Karina Ap. V. Grechi, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lúcia de Brito
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

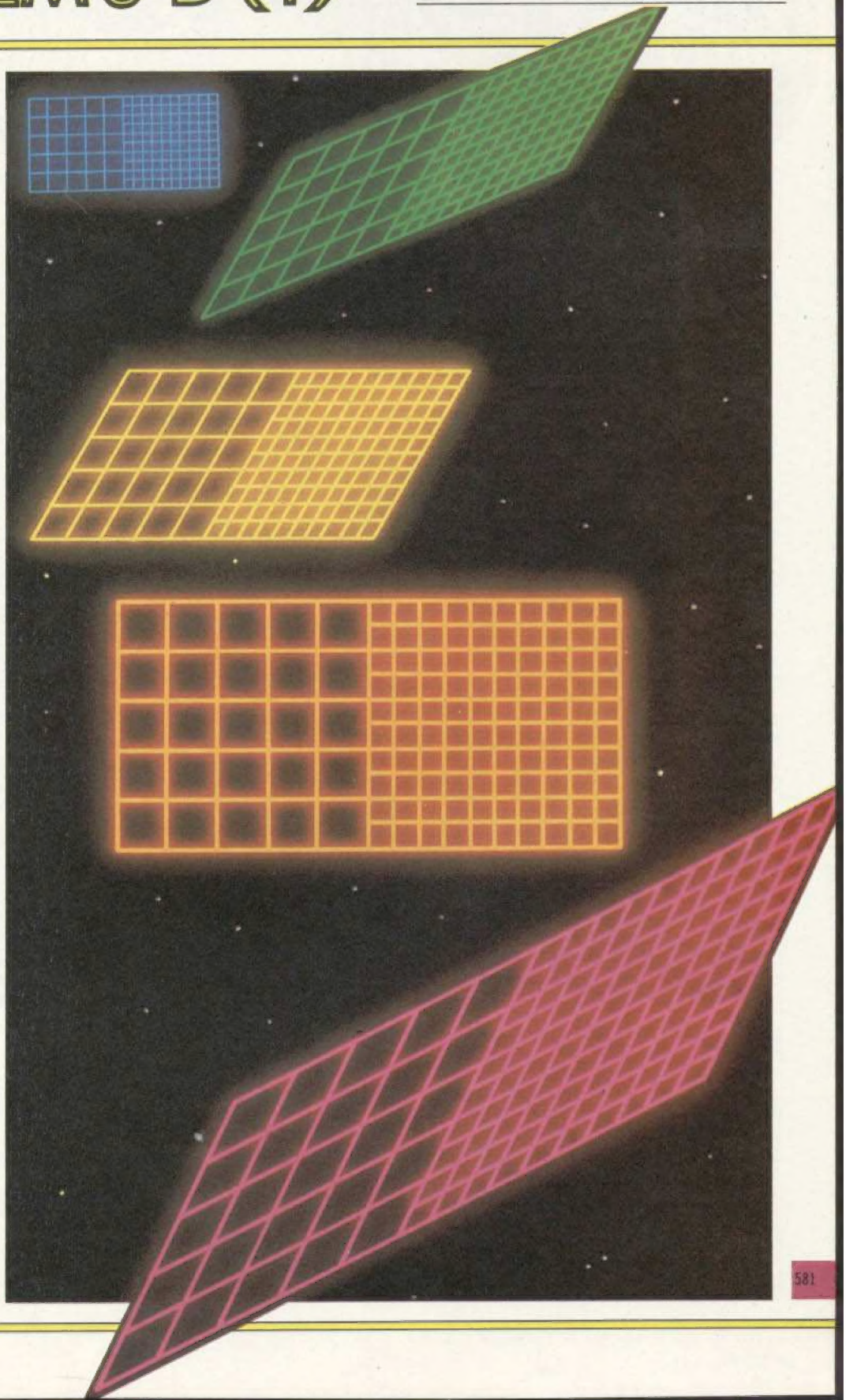
© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

PROGRAMAÇÃO DE GRÁFICOS EM 3-D (1)

- O QUE É UM DESENHO TIPO **WIREFRAME**?
- COMO ORGANIZAR AS ROTINAS
- DESENHE UMA GRADE
- DESENHE UM CÍRCULO

Neste primeiro artigo da série sobre desenho tipo wireframe, você aprenderá a elaborar grades e círculos. Mais tarde, verá como utilizá-los para estruturar imagens em 3-D.



Desenhos tridimensionais em movimento, do tipo *wireframe* (ou gradeado), têm se tornado imagens comuns em propagandas e, sem dúvida, impressionam muito. O efeito é uma das mais recentes contribuições do computador às comunicações visuais. Se, como usuário de um micro pessoal, você quiser produzir figuras como aquelas, certamente ficará desapontado ao verificar que é impossível obter o esplendor das imagens elaboradas por computadores, artistas e técnicos fotográficos nos comerciais de televisão.

Mas nem tudo está perdido: nesta série de artigos, você verá como extrair o máximo de seu computador, capacitando-o a produzir e movimentar imagens em 3-D de uma maneira que até engenheiros-desenhistas invejariam.

Como vimos em artigos anteriores, a possibilidade de endereçar pontos individuais (pixels) na tela proporciona ao micro uma poderosa ferramenta de desenho. O TK-90X, por exemplo, tem 256 pixels na horizontal e 176 na vertical. Normalmente, desenhamos linhas retas ou pontos que, combinados com recursos de cor, são usados para produzir imagens de alta resolução.

O método empregado para desenhar linhas em sistemas gráficos controlados por computador é, em geral, semelhante ao utilizado quando se trabalha com caneta e papel. Podemos mover o cursor pela tela fazendo com que ele deixe marcas ou não, e mudar de cor é tão simples quanto trocar de caneta.

A principal diferença entre desenhar com o computador ou manualmente está na rapidez da máquina e na perfeição com que ela traça linhas retas. Estas características tornam possível desenhar imagens em seus contornos ou mesmo wireframes tridimensionais.

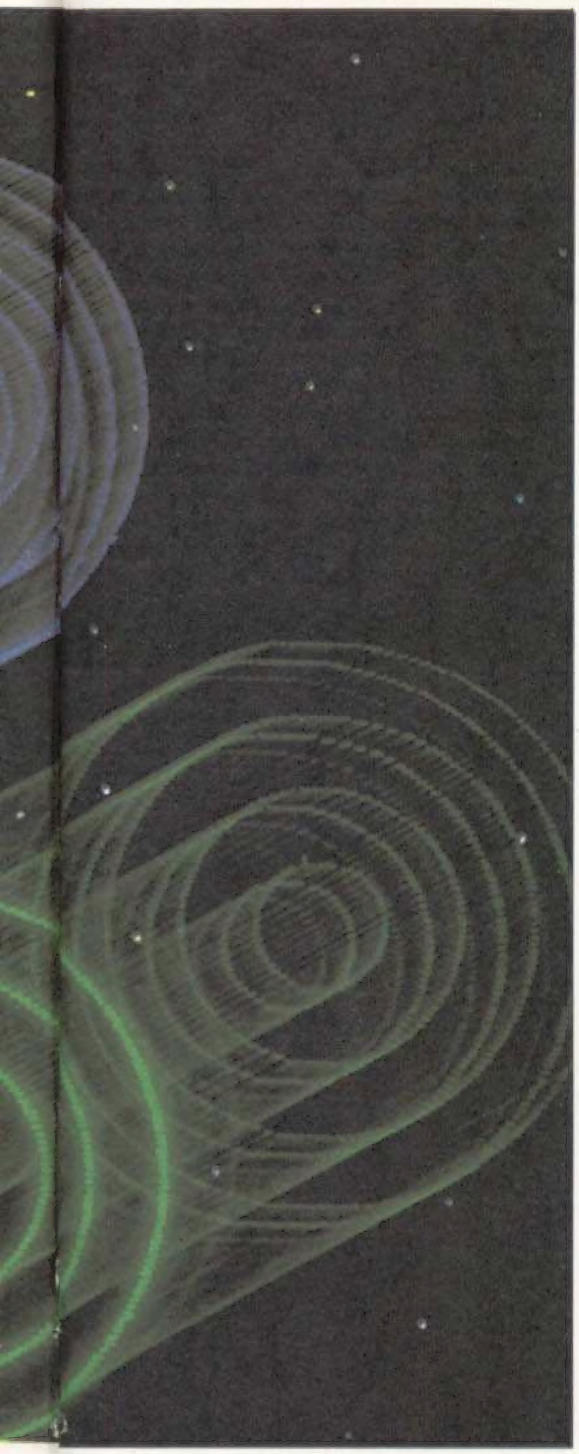
O QUE É, AFINAL, UM WIREFRAME?

Wireframes são representações imaginárias formadas por grades de linhas sobre a superfície do objeto. Geralmente, neste tipo de desenho, as linhas de fundo do objeto também ficam aparentes, pois, para escondê-las, precisaríamos de muitos cálculos extras. O resultado é uma figura que parece feita somente de uma armação de arame.

Os wireframes podem ser movimentados ou rodados, como vemos em diversas propagandas. Nos micros pessoais, não se consegue a mesma rapidez. Para alcançar velocidade semelhante, precisaríamos que os quadros mudassem pelo menos a cada 1/25 de segundo. Na verdade, a maioria dos computadores usados nos comerciais de TV também não são tão rápidos assim, mas as imagens parecem estar em "tempo real" graças a técnicas de filmagem.

Em desenhos estáticos, o tempo não é importante, embora cause irritação esperar demais até que uma figura se complete. Mas, por outro lado, muitas vezes é mais interessante ver a figura sendo desenhada do que vê-la acabada, sobretudo se ela for bem complexa.

Começaremos desenhando formas simples, como cubo ou esfera; à medida que nos familiarizarmos com as técnicas do wireframe, tentaremos formas mais complicadas. Neste artigo, apren-



deremos a criar figuras bidimensionais que parecerão estar em espaço tridimensional. Infelizmente, as rotinas aqui apresentadas não servem para os micros das linhas ZX-81 e TRS-80.

PRIMEIROS PASSOS

A execução de um desenho complexo do tipo wireframe é bem demorada, independentemente da velocidade ou da capacidade de traçar linhas de um micro. Os primeiros passos consistem na incorporação de um conjunto de rotinas, que incluem os comandos gráficos básicos. Precisamos de um comando que mova o cursor sem desenhar e outro que o movimente desenhando. Estes comandos variam de máquina para máquina, como é o caso do **MOVE**, **DRAW** e **LINE**, já vistos em outros artigos.

Um programa gráfico geralmente começa ajustando o computador para o modo gráfico, se necessário, e limpando a tela. Devemos optar sempre pela resolução de tela mais alta. Convém ainda estruturar o programa de modo que todos os comandos gráficos sejam coletados juntos em sub-rotinas. Com essa medida, evitaremos reescrever desnecessariamente porções de programa que realizam a mesma tarefa. Além disso, tanto a expansão quanto o entendimento do programa serão mais simples se os comandos que realizam certa tarefa forem coletados numa mesma área. A estruturação de um programa deixa-o, de fato, um pouco lento; isso, porém, não tem maior importância quando lidamos com figuras estáticas.

INICIALIZAÇÃO DA MÁQUINA

Para começar o trabalho com as rotinas de desenho, digite esta parte do programa, mas não a rode ainda pois está incompleta.

```
9000  REM   INICIO
9005 PI = 4 *  ATN (1)
9010  HGR2 : HCOLOR= 3:N = 0
9070  RETURN
9100  REM  MOVE
9110  HPLOT X1,Y1
9120  RETURN
9200  REM   DESENHA
9210  HPLOT X1,Y1 TO X2,Y2
9220  RETURN
```

```
9000 PCLS
9030 RETURN
```

```
9000 REM INICIO
9010 BORDER 4: PAPER 7: INK 0:
CLS : LET N=0
9070 RETURN
9100 REM MOVE
9110 PLOT X,Y
9120 RETURN
9200 REM DESENHA
9210 DRAW X-PEEK 23677,Y-PEEK 2
3678
9220 RETURN
```

```
9000 CLS
9030 RETURN
```

Os programas do TRS-Color e MSX são mais curtos porque esses computadores permitem que movimentemos o cursor e desenhemos usando um simples comando. Os programas do Spectrum e do Apple, por sua vez, contêm sub-rotinas que movimentam o cursor sem desenhar (linhas 9100 a 9120) e que desenham na tela (linhas 9200 a 9220). Essas rotinas combinadas formam uma única rotina de movimento e desenho.

Agora, para fazer um desenho, não precisaremos dizer ao computador como mover o cursor ou marcar a tela: bastará definir a forma do objeto nos termos dessas rotinas básicas.

DESENHO DE LINHAS

A forma mais simples que um desenho pode assumir é, sem dúvida alguma, a de uma linha. Aqui está a rotina que faz isso; digite-a, mas não rode o programa ainda.

```
9500  REM  LINHA
9510 X1 = XS:Y1 = YS: GOSUB 910
0
9520 X2 = XE:Y2 = YE: GOSUB 920
0
9550  RETURN
```

```
9500 LINE(XS,YS)-(XE,YE),PSET
9550 RETURN
```

```
9500 REM LINHA
9510 LET X=XS: LET Y=YS: GOSUB
9100
9520 LET X=XE: LET Y=YE: GOSUB
9200
9550 RETURN
```

```
9500 LINE(XS,YS)-(XE,YE),15
9550 RETURN
```

A rotina especifica uma posição inicial — coordenadas (XS, YS) — e uma posição final — coordenadas (XE, YE) — para a linha. Em alguns computadores é necessário utilizar os comandos **DRAW** ou **LINE**; em outros, isto já está especificado nas rotinas das linhas 9100 a 9220. A rotina que desenha linhas é a base da maioria dos programas do tipo wireframe.

O DESENHO DA GRADE

Para representar uma superfície e qualquer irregularidade que possa ter — como morros, vales, fendas etc. — é melhor visualizá-la não como uma área contínua dentro de um retângulo, mas como uma grade de linhas horizontais e verticais. Assim, todas as irregularidades poderão ser representadas por distorções dessas linhas.

A próxima seção do programa define a rotina que desenha uma grade. Digite-a, mas não a rode.

```
5000 JA = LW / NX
5010 XS = XA
5020  FOR J = 0 TO NX
5025 YS = YA:XE = XS:YE = YA +
LH
5030  GOSUB 9500
5040 XS = XS + JA
5050  NEXT J
5060 JA = LH / NY
5070 YS = YA + LH
5080  FOR J = 0 TO NY
5090 XS = XA + LW:XE = XA:YE =
YS
5100  GOSUB 9500
5110 YS = YS - JA
5120  NEXT J
5130  RETURN
```

```
5000 JA=LW/NX
5010 XS=XA
5020 FOR JB=0 TO NX
5025 YS=YA:XE=XS:YE=YA+LH
5030 GOSUB 9500
5040 XS=XS+JA
5050 NEXT JB
5060 JA=LH/NY
5070 YS=YA+LH
5080 FOR JB=0 TO NY
5090 XS=XA+LW:XE=XA:YE=YS
5100 GOSUB 9500
5110 YS=YS-JA
5120 NEXT JB
5130 RETURN
```

```
5000 LET JA=LW/NX
5010 LET XS=XA
5020 FOR J=0 TO NX
5025 LET YS=YA: LET XE=XS: LET
YE=YA+LH
5030 GOSUB 9500
5040 LET XS=XS+JA
5050 NEXT J
5060 LET JA=LH/NY
5070 LET YS=YA+LH
5080 FOR J=0 TO NY
5090 LET XS=XA+LW: LET XE=XA: L
ET YE=YS
5100 GOSUB 9500
5110 LET YS=YS-JA
5120 NEXT J
5130 RETURN
```

```
5000 JA=LW/NX
5010 XS=XA
5020 FOR JB=0 TO NX
5025 YS=YA:XE=XS:YE=YA+LH
5030 GOSUB 9500
5040 XS=XS+JA
5050 NEXT JB
5060 JA=LH/NY
5070 YS=YA+LH
5080 FOR JB=0 TO NY
5090 XS=XA+LW:XE=XA:YE=YS
5100 GOSUB 9500
5110 YS=YS-JA
5120 NEXT JB
5130 RETURN
```

As coordenadas (XA,YB) especificam o canto inferior esquerdo da grade. LW especifica a largura; LH, a altura; NX, o número de divisões horizontais, e NY, o número de divisões verticais. A variável JA determina a distância entre as linhas verticais, e o laço **FOR...NEXT** desenha horizontais, mantendo aquele intervalo. O segundo laço **FOR...NEXT** desenha verticais a intervalos calculados pela linha 5060.

A rotina entre as linhas 5000 e 5180 desenha as linhas horizontais da esquerda para a direita, e as verticais de baixo para cima, formando uma grade de superfície plana. Não poderíamos, portanto, representar irregularidades na superfície dessa grade.

Para mostrá-la na tela, digite as linhas seguintes, que chamam a rotina, e rode o programa.

```
100  GOSUB 9000
175 XA = 0:YA = 0:LW = 279:LH =
 191:NX = 16:NY = 14
180  GOSUB 5000
190  STOP
```

```
100 PMODE 4:SCREEN 1,1
105 PI=4*ATN(1)
110 GOSUB 9000
175 XA=0:YA=0:LW=255:LH=191:NX=
4:NY=3
180 GOSUB 5000
190 GOTO 190
```

```
100 GOSUB 9000
175 LET XA=0: LET YA=0: LET LW
=255: LET LH=175: LET NX=16:
LET NY=12
180 GOSUB 5000
190 STOP
```

```
100 SCREEN 2:COLOR 1,9,10
105 PI=4*ATN(1)
110 GOSUB 9000
175 XA=0:YA=0:LW=255:LH=191:NX=
4:NY=3
180 GOSUB 5000
190 GOTO 190
```

Ao rodar o programa, aparecerá uma grade ocupando toda a tela. Faça as mudanças abaixo para observar a versatilidade do programa:

```
175 XA = 10:YA = 10:LW = 240:LH
 = 144:NX = 1:NY = 1
```

```
175 XA=10:YA=10:LW=240:LH=160:N
X=1:NY=1
```

```
175 LET XA=10: LET YA=10: LET
LW=240: LET LH=144: LET NX=1:
LET NY=1
```

```
175 XA=10:YA=10:LW=240:LH=160:N
X=1:NY=1OK
```

Desta vez, aparece na tela uma caixa retangular, já que se especificou uma grade com apenas uma divisão horizontal e uma vertical. Fornecendo valores adequados a NX e NY, como acima, podemos construir uma grade com números de linhas horizontais diferente do de verticais. Faça as mudanças que se seguem e rode o programa:

```
175 XA = 0:YA = 0:LW = 180:LH =
 130:NX = 16:NY = 16
```

```
175 XA=0:YA=31:LW=160:LH=160:NX
=15:NY=10
```

```
175 LET XA=0: LET YA=0: LET LW
=160: LET LH=144: LET NX=15:
LET NY=10
```

```
175 XA=0:YA=31:LW=160:LH=160:NX
=15:NY=10
```

Para a forma quadrada, determinamos os valores apropriados de LW e LH e o número de divisões por NX e NY.

O DESENHO DE CÍRCULOS

Os círculos também são muito úteis nos desenhos tipo wireframe. Em alguns

micros, eles podem ser executados diretamente por um comando, bastando que se definam o centro e o raio.

O comando direto, contudo, não nos oferece o grau de controle necessário para a elaboração de um desenho tridimensional. Em perspectiva, um círculo visto de um certo ângulo pode parecer uma elipse ou até uma curva. Embora desenhe elipses, o comando **CIRCLE** não é capaz de lhes conferir a tridimensionalidade essencial para garantir a aparência realística ao objeto. Convém, portanto, definir uma função genérica.

Podemos compor um círculo usando uma série de segmentos de reta. Se estes forem suficientemente pequenos, a curva parecerá contínua. Mas, quanto menores forem os segmentos de reta, maior será o número deles e mais prolongado o tempo de execução. Aqui está uma rotina que desenha um círculo de raio R com centro em (XS, YS). Digite-a, mas não a rode ainda.

```
6000  IF N = 0 THEN N = 20 +  I
NT (R / 10)
6020 JA = 2 * PI / N
6050 XR = XS:YR = YS
6060 JB = 0:XS = XS + R
6070  FOR J = 2 TO N
6080 JB = JB + JA
6090 XE = XR + R *  COS (JB):YE
 = YR + R *  SIN (JB): GOSUB 95
00
6100 XS = XE:YS = YE
6110  NEXT J
6120 XE = XR + R:YE = YR: GOSUB
 9500
6130 XS = XR:YS = YR
6160  RETURN
```

```
6000 IF N=0 THEN N=20+INT(R/10)
6020 JA=2*PI/N
6050 XR=XS:YR=YS
6060 JB=0:XS=XS+R
6070 FOR JC=2 TO N
6080 JB=JB+JA
6090 XE=XR+R*COS(JB):YE=YR+R*SI
N(JB):GOSUB 9500
6100 XS=XE:YS=YE
6110 NEXT JC
6120 XE=XR+R:YE=YR:GOSUB 9500
6130 XS=XR:YS=YR
6160 RETURN
```

```
6000 IF N=0 THEN  LET N=20+INT
(R/10)
6020 LET JA=2*PI/N
6050 LET XR=XS: LET YR=YS
6060 LET JB=0: LET XS=XS+R
6070 FOR J=2 TO N
6080 LET JB=JB+JA
6090 LET XE=XR+R*COS JB: LET YE
=YR+R*SIN JB: GOSUB 9500
6100 LET XS=XE: LET YS=YE
6110 NEXT J
6120 LET XE=XR+R: LET YE=YR: GO
SUB 9500
6130 LET XS=XR: LET YS=YR
6160 RETURN
```

A variável N especifica o número de segmentos de reta que serão usados na composição do círculo. Caso façamos N = 0, a linha 6000 calculará o número de segmentos necessários para fazer o círculo mais liso, levando em conta, é claro, o tamanho dele.

A linha 6020 calcula o ângulo de cada segmento de reta na circunferência. A linha 6050 move o cursor para uma posição na circunferência. O laço **FOR...NEXT** traça todos os segmentos, com exceção do último, que é desenhado pela linha 6120, para garantir que se una ao primeiro.

Para ver a rotina funcionando, apague a linha 180 e acrescente estas:

```
150  FOR R = 20 TO 70 STEP 10
155 XS = 128:YS = 102:N = 24
160  GOSUB 6000
170  NEXT R
```

```
150 FOR R=0 TO 100 STEP 20
155 XS=128:YS=102:N=24
160 GOSUB 6000
170 NEXT R
```

```
150 FOR R=20 TO 70 STEP 10
155 LET XS=128: LET YS=102:
LET N=24
160 GOSUB 6000
170 NEXT R
```

```
150 FOR R=0 TO 100 STEP 20
155 XS=128:YS=102:N=24
160 GOSUB 6000
170 NEXT R
```

Posso colorir wireframes?

Desenhos do tipo wireframe geralmente são apresentados em duas cores, sobretudo preto e branco.

A presença de muitas cores tende a complicar a imagem, muitas vezes anulando o efeito tridimensional. Além disso, a adição de cores a um desenho complexo esbarra em limitações da própria máquina.

No TRS-Color, o maior número de cores sacrifica a alta resolução gráfica, pois a resolução se reduz à metade quando passamos de um modo de duas cores para um de quatro.

No Spectrum e no MSX, determinadas porções da tela só podem apresentar duas cores. No Spectrum, esta porção corresponde a cada bloco de oito por oito pontos da tela; no MSX, a cada conjunto de oito pontos adjacentes na horizontal.

Já no Apple e no TK-2000, algumas cores ocupam colunas pares, outras ímpares, em alta resolução. Se colocarmos pontos ou linhas em colunas inadequadas à sua cor, eles simplesmente não aparecerão na tela.

Rodado o programa, um certo número de círculos aparecerá no centro da tela. Como na rotina da grade, podemos variar os parâmetros no programa que chama a rotina e obter uma disposição de círculos diferente. A linha 150 ajusta o raio do primeiro círculo e determina o quanto ele aumenta. A 155 define o centro do círculo e o número de segmentos de reta que formarão a circunferência. Como exercício, varie esses valores e veja os resultados.

Talvez você queira saber o que aconteceu com a rotina da grade: ela ainda está na memória, mas, como reescrevemos as linhas de código que chamam a rotina, o computador não as mostra. Rotinas como esta podem ser guardadas num arquivo de rotinas gráficas para uso posterior. Elas terão utilidade em vários tipos de programa gráfico, bem como nos de wireframes. Na medida em que for preciso, adicione novas rotinas. Uma vez no computador, salve-as em cassete ou disco e carregue-as novamente na memória quando for utilizá-las. Junte-as, se necessário. No próximo artigo desta série, veremos como usar essas rotinas na criação de wireframes tridimensionais.

UM EDITOR DE TEXTOS (2)

Adicionar, corrigir, apagar, tudo é possível com um programa editor. Portanto, se você quer obter um texto perfeito, comece com um rascunho e melhore-o à vontade.

Neste artigo, você encontrará a parte principal do programa editor de textos e as explicações sobre o uso das funções de edição em cada máquina. Embora os procedimentos sejam muito parecidos, controles específicos e algumas características variam.

Após digitar as linhas aqui apresentadas, você já poderá usar todas as funções de edição do programa. Mas lembre-se de que só será possível imprimir o texto criado depois da última parte, que daremos no próximo artigo.

Como você verá, as opções de edição oferecidas pelo programa são úteis em vários níveis. O mais simples consiste na correção dos erros de grafia. É facílimo eliminá-los: basta colocar sua "caneta eletrônica", o cursor, sobre a palavra errada e inserir ou apagar as letras que quiser.

Se a ortografia não é um problema para você, mas sim a composição do texto, este programa se revelará ideal. Ao escrever algo importante, como uma solicitação de emprego ou uma explicação ao gerente sobre um problema em sua conta bancária, você já não precisará gastar uma pilha de folhas de papel, riscando e reescrevendo até encontrar o termo mais preciso, ou a forma mais adequada.

Vá direto ao computador e economize papel e paciência.

Comece digitando um rascunho do texto e, depois, melhore-o. Analise o que já escreveu "passeando" pelo texto, do início ao fim, quantas vezes julgar necessário. Quando decidir fazer alguma modificação, corrija, apague ou introduza o que quiser. Trabalhe à vontade, até que o texto lhe agrade.

Uma outra possibilidade que o programa lhe oferece é a de armazenar o texto elaborado, para usá-lo em outra oportunidade ou, apenas, para saber mais tarde o que escreveu.

Os recursos disponíveis neste programa são semelhantes, embora muito mais simples, aos dos mais modernos processadores de texto usados, por exemplo, para produzir o que você está lendo agora. Quem já viu como tudo isso era feito há pouco tempo (e ainda é, em alguns lugares) não terá dúvidas a respeito das vantagens de utilizar um programa editor.

O programa do Spectrum foi elaborado para uso com um gravador cassete. Se você tem um Microdrive, faça as modificações necessárias.

As funções de edição são quase idênticas às do BASIC e, portanto, não é necessário um editor especial. Todas as entradas de texto, bem como as alterações, devem ser feitas na parte inferior da tela, a área de trabalho, ficando a parte superior para a visualização do que já foi escrito.

Durante a edição de linhas na área de trabalho, pressione **<CAPS SHIFT>** e 5 para mover o cursor para a esquerda e **<CAPS SHIFT>** e 8 para movê-lo para a direita. Quando ele estiver na posição adequada, digite os caracteres necessários. Para apagar, coloque o cursor à direita do caractere e pressione **<CAPS SHIFT>** e 0.

Para transferir o texto para a memória e para a área superior, tecle **<ENTER>**. Linhas com mais de 64 caracteres irão automaticamente para a memória, visto que a área de trabalho su-

- CORREÇÃO DE ERROS ORTOGRÁFICOS
- COMPOSIÇÃO DO RASCUNHO AO TEXTO FINAL
- ANALISE SEM PRESSA
- ARMAZENAGEM DO TEXTO
- AJUSTE DO TAMANHO E ACERTO DOS PARÁGRAFOS
- AS FUNÇÕES DE EDIÇÃO EM CADA MÁQUINA

porta no máximo duas linhas de texto. Mas elas permanecerão juntas para efeito de impressão, a não ser que se usem comandos de formatação.

As teclas de cursor para cima e para baixo são usadas para localizar linhas do texto na memória. O marcador fica abaixo da linha de interesse, na área de visualização do texto. Esta linha pode ser copiada para a área de trabalho usando-se a função **EDIT** habitual — pressione <**SHIFT**> e 1. Para apagar a linha, pressione <**CAPS SHIFT**> e 9, deixando o modo editor e, em seguida, <**CAPS SHIFT**> e <**SYMBOL SHIFT**> simultaneamente.

Para ler todo o texto da memória, use as teclas de cursor para cima e para baixo ou <**CAPS SHIFT**> e 6 e <**CAPS SHIFT**> e 7. Estas teclas moverão o texto para cima ou para baixo, uma linha de cada vez.

```
1000 REM imprime a tela
1005 PLOT 0,13: DRAW 255,0: PLO
T 0,14: DRAW 255,0
1010 PRINT AT 0,0;: FOR n=p-10
TO p+8
1020 IF n<1 OR n>200 THEN  PRIN
T s$: GOTO 1050
1025 IF n=p THEN  PRINT
1030 PRINT t$(n)
1040 POKE 22528+320,120
1050 NEXT n
1060 RETURN
2000 REM entrada
2010 LET i$="": LET j$=""
2015 PRINT #1;AT 0,0;i$; FLASH
1; BRIGHT 1;" "; FLASH 0; BRIGH
T 0;j$;" "
2020 PAUSE 0: LET a$=INKEY$: IF
 a$="" THEN  GOTO 2020
2025 SOUND .01,20
2030 IF a$<CHR$ 32 THEN  GOSUB
2500
2040 IF a$>CHR$ 31 AND a$<CHR$
123 THEN  LET i$=i$+a$
2042 IF a$=CHR$ 13 AND b=ext-6
THEN  PRINT #1;AT 0,0;s$;s$; FL
ASH 1;"ARQUIVO LOTADO": SOUND 2
,10: RETURN
2045 IF a$=CHR$ 13 OR LEN i$+LE
N j$=64 THEN  PRINT #1;AT 0,0;s
$;s$;s$: LET i$=i$+j$: GOTO 210
0
2050 IF a$=CHR$ 14 THEN  RETURN

2052 IF a$=CHR$ 6 THEN  INPUT "
Introduza palavra procurada", L
INE z$: IF z$="" THEN  GOTO 205
2
2053 IF a$=CHR$ 6 THEN  LET p=4
: GOSUB 8000
2054 IF a$=CHR$ 4 THEN  GOSUB 8
000
2055 IF a$=CHR$ 5 THEN  GOSUB 8
500
2060 GOTO 2015
2100 IF LEN i$>32 THEN  GOTO 21
50
2105 FOR n=b+1 TO p STEP -1
2110 LET t$(n+1)=t$(n)
2120 NEXT n
2130 LET t$(n+1)=i$: LET p=p+1:
 LET b=b+1
2140 GOSUB 1000: GOSUB 2500: GO
TO 2000
2150 FOR n=b+1 TO p STEP -1
2160 LET t$(n+2)=t$(n): LET t$(
n+3)=t$(n+1)
2170 NEXT n
2180 LET t$(n+1)=i$( TO 32): LE
T t$(n+2)=i$(33 TO ): LET p=p+2
: LET b=b+2
2190 GOTO 2140
2200 LET p=p-1: FOR n=p TO b+1
2210 LET t$(n)=t$(n+1)
2220 NEXT n
2225 LET b=b-1
2230 GOSUB 1000
2240 RETURN
2500 REM codigos de controle
2520 IF a$=CHR$ 10 AND p<b-2 TH
EN  LET p=p+1: GOSUB 1000
2530 IF a$=CHR$ 11 AND p>t+3 TH
EN  LET p=p-1: GOSUB 1000
2540 IF a$=CHR$ 12 AND LEN i$>0
 THEN  LET i$=i$( TO LEN i$-1)
2550 IF a$=CHR$ 8 AND LEN i$>0
THEN  LET j$=i$(LEN i$)+j$: LET
 i$=i$( TO LEN i$-1)
2560 IF a$=CHR$ 9 AND LEN j$>0
THEN  LET i$=i$+j$(1): LET j$=j
$(2 TO )
2570 IF a$<>CHR$ 7 THEN  GOTO 2
580
2572 LET j$=t$(p-1): LET i$="":
 PRINT #1;AT 0,0;s$;s$
2575 IF j$(LEN j$)=CHR$ 32 THEN
  LET j$=j$( TO LEN j$-1): IF L
EN j$>0 THEN  GOTO 2575
2580 IF a$=CHR$ 15 AND p>4 THEN
  GOSUB 2200
2690 RETURN
3000 REM cores
3010 PRINT AT 10,4;"Selecione c
or de fundo (0-7)"
3020 PAUSE 0: LET a$=INKEY$: IF
 a$<"0" OR a$>"7" THEN  GOTO 30
20
3030 PAPER VAL a$: BORDER VAL a
$: CLS : RETURN
```

Em geral, o texto é exibido na tela (e eventualmente impresso) em letras maiúsculas. Minúsculas podem ser usadas teclando <**SHIFT**>0. Na tela, aparecerão como caracteres invertidos (fundo escuro, caractere claro).

Os controles de edição seguem as diretrizes anteriormente mencionadas. Edita-se o texto na área de trabalho, localizada na parte inferior da tela. A parte superior é utilizada para mostrar o

texto que já está na memória. As teclas de controle do cursor têm um papel importante na edição. As teclas <→> e <←> permitem que você mova o cursor pelo texto na área de trabalho. Se você pressionar <**SHIFT**> juntamente com <←>, o cursor pulará para o início ou fim da linha.

Para inserir caracteres em um ponto da linha na área de trabalho, coloque o cursor à direita do local escolhido e tecle o que desejar. Não se pode sobrepor caracteres. Para eliminar erros, coloque o cursor sobre o caractere e tecle a seta para baixo.

Para ativar o modo editor, tecle a seta para cima. O cursor vai para a última posição acessada, indicada por um sinal > piscando.

No modo editor, pode-se inspecionar o texto da memória, rolando-o para cima e para baixo com as setas respectivas. Avanços (ou retrocessos maiores) são possíveis com as teclas U e D, que pulam de 10 em 10 linhas.

Se quiser apagar linhas de texto, posicione o marcador logo abaixo da linha e pressione <**SHIFT**> e seta para baixo, ao mesmo tempo. Linhas em branco podem ser inseridas a partir da área de trabalho, pressionando-se <**ENTER**> quando ela está vazia.

Para copiar uma linha da memória para a área de trabalho, coloque o marcador logo abaixo dela e pressione **C** (dentro do modo editor). A linha modificada não substitui a original, devendo esta ser apagada depois. Volta-se do modo editor para a área de trabalho teclando-se <**ENTER**>. <**CLEAR**> retorna da área de trabalho para o menu de edição.

```
2500 PM=5:IF CP<5 THEN PM=CP
2510 TB$=INKEY$:IF TB$="" THEN
PRINT @PM*32,"":PRINT @PM*32,">
":GOTO 2510
2520 CP=CP+(TB$="^")-(TB$=CHR$(
10))+10*((TB$="U")-(TB$="D"))
2530 IF CP<1 THEN CP=1
2540 IF CP>TL THEN CP=TL
2550 GOSUB 2090
2560 IF TB$=CHR$(13) THEN RETUR
N
2570 IF CP>1 AND TB$=CHR$(91) T
HEN TL=TL-1:FOR K=(CP-1) TO TL:
TX$(K)=TX$(K+1):NEXT:TX$(TL+1)=
"":CP=CP-1:GOSUB 2090
2580 IF CP>1 AND TB$="C" THEN F
OR K=32 TO 1 STEP -1:IF MID$(TX
$(CP-1),K,1)="" THEN NEXT ELSE
A$=LEFT$(TX$(CP-1),K)+"":RETURN
2590 IF TB$="P" GOSUB 5070
2600 IF TB$="@" THEN SF=SF+1:IF
 SF=1 THEN SS=CP ELSE SE=CP:SF=
0:GOSUB 5130
2610 GOTO 2500
3000 RETURN 'LINHA TEMPORARIA
4000 CLS:IF TL=1 THEN PRINT @7,
"nada para guardar"FOR Z=1 TO 1
000:NEXT:RETURN
4010 CLS:LINEINPUT" NOME DO ARQ
UIVO?";F$
4020 IF LEFT$(F$,1)<"A" OR LEFT
$(F$,1)>"Z" THEN 4010
4030 IF TS=1 THEN 4120
4040 CLS:MOTORON:AUDIO ON:PRINT
"POSICIONE O TAPE E PRESSIONE
  <ENTER>"
4050 IF INKEY$<>CHR$(13) THEN 4
050 ELSE MOTOROFF:AUDIO OFF:PRI
NT"POSICIONE O GRAVADOR EM 'REC
' E PRESSIONE <ENTER>"
4060 IF INKEY$<>CHR$(13) THEN 4
060
4070 MOTORON:FOR K=1 TO 1000:NE
XT:OPEN"O",#-1,F$
4080 PRINT #-1,CP,TL
4090 FOR K=1 TO TL-1:PRINT #-1,
TX$(K):NEXT
4100 CLOSE #-1
4110 RETURN
4120 CLS:PRINT"CERTIFIQUE-SE DE
 QUE O DRIVE    ESTA LIGADO E O
 DISCO INSERIDO. PRESSIONE <ENT
ER>PARA CONTINUAR"
4130 IF INKEY$<>CHR$(13) THEN 4
130 ELSE F$=F$+"/DAT"
4140 OPEN  "O", #1, F$
4150 WRITE #1,CP,TL
4160 FOR K=1 TO TL-1
4170 WRITE #1,TX$(K)
4180 NEXT:CLOSE #1:RETURN
4500 CLS:PRINT @8,BL$;"carregar
";BL$;"um";BL$;"arquivo";BL$:IF
 TL=1 THEN 4540
4510 PRINT "VOCE TEM CERTEZA (S
/N)?"
4520 R$=INKEY$:IF R$<>"S" AND R
$<>"N" THEN 4520
4530 IF R$="N" THEN RETURN
4540 CLS:LINEINPUT"NOME DO ARQU
IVO: ";F$
4550 IF LEFT$(F$,1)< "A" OR LEF
T$(F$,1)>"Z" THEN 4540
4560 IF DL=1 THEN 4645
4570 MOTORON:AUDIOON:PRINT"POSI
CIONE O TAPE EM 'PLAY' E    PRE
SSIONE <ENTER>"
4580 IF INKEY$<>CHR$(13) THEN 4
580
4590 OPEN "I",#-1,F$
4600 INPUT #-1,CP,TL
4610 FOR K=1 TO TL-1:INPUT #-1,
TX$(K):NEXT
4620 CLOSE #-1:GOSUB 2090
4630 TX$(TL)=STRING$(32,126)
4640 RETURN
4645 F$=F$+"/DAT"
4650 OPEN "I", #1, F$:INPUT #1,
CP,TL
4660 FOR K=1 TO TL-1:INPUT #1,T
X$(K)
4670 NEXT:CLOSE #1:RETURN
5000 CLS:PRINT @9,BL$;"selecao"
;BL$;"e";CHR$(124);"s";BL$:PRIN
T @96,"CARREGAR DE (F)ITA OU (D
)ISCO?";
5010 B$=INKEY$:IF B$<>"F"  AND
B$<>"D" THEN 5010
5020 PRINT B$:DL=0:IF B$="D" TH
EN DL=1
5030 PRINT:PRINT"GRAVAR EM (F)I
TA OU (D)ISCO?";
5040 B$=INKEY$:IF B$<>"F" AND B
$<>"D" THEN 5040
5050 PRINT B$:TS=0:IF B$="D" TH
EN TS=1
5060 RETURN
5070 RETURN 'LINHA TEMPORARIA
5130 RETURN 'LINHA TEMPORARIA
5500 CLS:PRINT @3,BL$;"preparac
ao";BL$;"da";BL$;"impressora";B
L$
5510 PRINT @128,;:INPUT"LARGURA
 DO FORMULARIO";MW:MW=INT(MW):I
F MW<1 THEN 5510
5520 INPUT "COMPRIMENTO DE LINH
A";TW:TW=INT(TW):IF TW<1 OR TW>
MW THEN 5520
5530 INPUT"COMPRIMENTO DO FORMU
LARIO";PL:PL=INT(PL):IF PL<1 TH
EN 5530
5540 INPUT"LINHAS POR PAGINA";T
H:TH=INT(TH):IF TH<1 OR TH>PL T
HEN 5530
5550 GP=INT((MW-TW)/2):LF$=STRI
NG$(INT((PL-TH)/2),13)
5560 PRINT:PRINT:PRINT" CONFIRM
A (S/N)?"
5570 R$=INKEY$:IF R$<>"N" AND R
$<>"S" THEN 5570
5580 IF R$="S" THEN RETURN ELSE
 5500
```

Os editores para o MSX e o Apple seguem, em linhas gerais, o que já foi dito. É importante prestar atenção quando se usam minúsculas, pois as respostas requeridas pelo computador só são aceitas em maiúsculas. Por outro lado, dentro do editor, podem-se usar sem problemas os sinais de pontuação e acentuação (esta, no caso do MSX).

Trabalha-se o texto na área situada na parte inferior da tela. Quando se entra nesta área, o cursor é visível no começo na linha. A partir daí, escreve-se normalmente. As setas para a direita e para a esquerda movimentam o cursor nas respectivas direções. Pode-se acelerar sua movimentação por meio de <**CTRL**> <**A**> e <**CTRL**> <**S**>. A pressão simultânea dessas teclas coloca o cursor na primeira e última posição da linha, respectivamente. Para inserir um caractere, leve o cursor para a posição imediatamente à direita de onde fará a inserção e digite o caractere. Não é possível sobrepor caracteres. Para apagar algo errado, coloque o cursor sobre o caractere e digite <**CTRL**> <**D**>.

Para ativar o modo editor, pressione <**CTRL**> <**E**>. O marcador do texto começará a piscar. Teclando <**CTRL**> <**O**> e <**CTRL**> <**Z**>, você poderá inspecionar o texto todo. Esses comandos fazem com que o marcador (e o texto) ro-

lem para cima ou para baixo. <**CTRL**> <**W**> e <**CTRL**> <**X**> provocam saltos maiores (de 10 linhas).

Para apagar linhas de texto, estando no modo editor, pressiona-se <**CTRL**> <**L**>. A linha logo acima do marcador será apagada.

Quando quiser transferir uma linha de memória para a área de trabalho, tecle <**CTRL**> <**C**>, com o marcador embaixo da linha desejada. A linha aparecerá na área de trabalho e poderá ser editada imediatamente. Note que, ao teclar <**RETURN**> para devolvê-la para a memória, ela não substituirá a original. É necessário apagá-la.

Linhas em branco podem ser criadas simplesmente teclando-se <**RETURN**>, com a área de trabalho vazia.

O programa para o Apple armazena e grava dados somente em disco. Já o programa para o MSX trabalha apenas com gravador cassete. Não estranhe se a leitura de dados no Apple for morosa. Para não haver problema com os sinais de pontuação, é necessário que os caracteres sejam lidos um de cada vez. Se você quiser mais velocidade, não se importando com a pontuação, elimine o laço das linhas 4600 e 4610 e deixe apenas a instrução **INPUT TX$ (K)**.

```
2500 PM=9:IF CP<9 THEN PM=CP
2510 TB$=INKEY$:IF TB$=""THEN L
OCATE 0,PM:PRINTCHR$(219);:PRIN
TCHR$(8);CHR$(175);:GOTO 2510
2520 CP=CP+(TB$=CHR$(17))-(TB$=
CHR$(26))+10*((TB$=CHR$(23))-(T
B$=CHR$(24)))
2530 IF CP<1 THEN CP=1
2540 IF CP>TL THEN CP=TL
2550 GOSUB 2080
2560 IF TB$=CHR$(27) THEN RETUR
N
2570 IF CP>1 AND TB$=CHR$(12) T
HEN TL=TL-1:FOR K=CP-1 TO TL:TX
$(K)=TX$(K+1):NEXT:TX$(TL+1)=""
:CP=CP-1:GOSUB 2080
2580 IF CP>1 AND TB$=CHR$(3) TH
EN A$=TX$(CP-1)+" ":RETURN
2590 IF TB$=CHR$(16) THEN GOSUB
 5000
2600 IF TB$=CHR$(15) THEN SF=SF
+1:IF SF=1 THEN SS=CP ELSE SE=C
P:SF=0:GOSUB 5060
2610 GOTO 2500
3000 RETURN 'LINHA TEMPORARIA
4000 CLS:BL$="Gravar na fita":G
OSUB 220:IF TL=1 THEN BEEP:LOCA
TE 12,11:PRINT"Nada a gravar!":
FOR Z=1 TO 1000:NEXT:RETURN
4010 LOCATE 4,4:LINEINPUT"Nome
do arquivo? ";F$
4020 IF LEFT$(F$,1)<"A" OR LEFT
$(F$,1)>"Z"THEN 4010
4030 LOCATE 1,6:PRINT"Posicione
 a fita e pressione <RETURN>"
4040 IF INKEY$<>CHR$(13) THEN 4
040 ELSE PRINT:PRINT"Deixe o gr
avador pronto para iniciar    g
ravação e pressione <RETURN>"
4050 IF INKEY$<>CHR$(13) THEN 4
050
4060 OPEN F$ FOR OUTPUT AS #1
4070 PRINT#1,CP,TL
4080 FOR K=1 TO TL-1:PRINT#1,TX
$(K):NEXT
4090 CLOSE #1
4100 RETURN
4500 CLS:BL$="Carregar da fita"
:GOSUB 220:IF TL=1 THEN 4540
4510 LOCATE 12,12:PRINT"Confirm
e (S/N) ";
```

```
4520 R$=INKEY$:IF R$="" THEN 45
20
4530 IF R$<>"S" THEN RETURN ELS
E TL=1:GOTO 4500
4540 LOCATE 4,4:LINEINPUT"Nome
do arquivo? ";F$
4550 IF LEFT$(F$,1)<"A" OR LEFT
$(F$,1)>"Z" THEN 4540
4560 PRINT:PRINT:PRINTTAB(2);"P
osicione a fita e tecle <RETURN
>"
4570 IF INKEY$<>CHR$(13)THEN457
0
4580 PRINT:PRINT"Deixe o gravad
or pronto para leitura e   tecl
e <RETURN>"
4590 IF INKEY$<>CHR$(13)THEN459
0
4600 OPEN F$ FOR INPUT AS #1
4610 INPUT #1,CP,TL
4620 FOR K=1 TO TL-1:LINEINPUT
#1,TX$(K):NEXT
4630 CLOSE #1:RETURN
5500 CLS:BL$="Configura impress
ora":GOSUB 220
5510 LOCATE 0,5:PRINT"Configura
ção atual:"
5520 PRINT:PRINT:PRINT"Largura
da linha: ";MW;" col"
5530 PRINT"Largura do texto: ";
TW;" col"
5540 PRINT:PRINT"Comprimento da
 pag.: ";PL;" linhas"
5550 PRINT"Comprimento do texto
: ";TH;" linhas"
5560 PRINT:PRINT:PRINT"Quer alt
erar? ";
5570 R$=INKEY$:IF R$="" THEN 55
70
5580 IF R$<>"S" THEN RETURN
5590 CLS:GOSUB 220:LOCATE 0,8
5600 INPUT"Largura da linha ";M
W
5610 MW=INT(MW):IF MW<1 THEN 56
00
5620 INPUT"Largura do texto ";T
W
5630 TW=INT(TW):IF TW<1 OR TW>M
W THEN 5620
5640 PRINT:INPUT"Comprimento da
 pag. ";PL
5650 PL=INT(PL):IF PL<1 THEN 56
40
5660 INPUT"Comprimento do texto
 ";TH
5670 TH=INT(TH):IF TH<1 OR TH>P
L THEN 5660
5680 GP=INT((MW-TW)/2):LF=INT((
PL-TH)/2):LF$=STRING$(LF,13)
5690 GOTO 5500
```

```
2500 PM = 9: IF CP < 9 THEN PM
= CP
2510  HTAB 1: VTAB PM + 1: PRIN
T ">"; CHR$ (8);: GET TB$
2520 CP = CP + (TB$ =  CHR$ (26
)) - (TB$ =  CHR$ (17)) + 10 *
((TB$ =  CHR$ (24)) - (TB$ =  C
HR$ (23)))
2530  IF CP < 1 THEN CP = 1
2540  IF CP > TL THEN CP = TL
2550  GOSUB 2090
2560  IF TB$ =  CHR$ (27) THEN
 RETURN
2570  IF CP > 1 AND TB$ =  CHR$
 (12) THEN TL = TL - 1: FOR K =
 CP - 1 TO TL:TX$(K) = TX$(K +
1): NEXT :TX$(TL + 1) = "":CP =
 CP - 1: GOSUB 2090
2580  IF CP > 1 AND TB$ =  CHR$
 (3) THEN A$ = " " + TX$(CP - 1
) + " ": RETURN
2590  IF TB$ =  CHR$ (16) THEN
 GOSUB 5200
2600  IF TB$ =  CHR$ (15) THEN
SF = SF + 1: IF SF = 1 THEN SS
= CP: GOTO 2620
2610  IF TB$ =  CHR$ (15) THEN
SE = CP:SF = 0: GOSUB 5300
2620  GOTO 2500
3000  RETURN : REM  LINHA TEMPO
RARIA
```

```
4000  HOME :BL$ = "GRAVAR": GOS
UB 250
4010  IF TL = 1 THEN  PRINT  CH
R$ (7);: HTAB 13: VTAB 12: PRIN
T "NADA A GRAVAR!": FOR Z = 1 T
O 2000: NEXT : RETURN
4020  HTAB 5: VTAB 5: PRINT "NO
ME DO ARQUIVO? (MAX 20 CARACT)"
: INPUT "->";F$
4030  IF  ASC (F$) < 65 OR  ASC
 (F$) > 90 THEN 4000
4040 F$ =  LEFT$ (F$,20):F$ = F
$ + ".TXT"
4050  PRINT D$;"OPEN";F$;",S";S
1;",D";S2
4060  PRINT D$;"DELETE";F$
4070  PRINT D$;"OPEN";F$
4080  PRINT D$;"WRITE";F$
4090  PRINT CP: PRINT TL
4100  FOR K = 1 TO TL - 1: PRIN
```

```
T TX$(K): NEXT
4110  PRINT D$;"CLOSE": RETURN
4500  HOME :BL$ = "CARREGAR": G
OSUB 250
4510  HTAB 5: VTAB 5: PRINT "TE
CLE [CR] PARA RETORNAR AO MENU"
: HTAB 5: PRINT "NOME DO ARQUIV
O?": INPUT "->";F$
4520  IF F$ = "" THEN  RETURN
4530  IF  ASC (F$) < 65 OR  ASC
 (F$) > 90 THEN 4500
4540 F$ = F$ + ".TXT"
4550  PRINT D$;"OPEN";F$;",S";L
1;",D";L2
4560  PRINT D$;"READ";F$
4570  INPUT CP,TL
4580  FOR K = 1 TO TL - 1
4590 TX$(K) = ""
4600  GET B$: IF B$ =  CHR$ (13
5020  VTAB 8: PRINT "CONFIGURAC
AO ATUAL:"
5030  PRINT : PRINT  TAB( 5)"CA
RREGA DE ": PRINT  TAB( 10)"SLO
) THEN  NEXT : GOTO 4620
4610 TX$(K) = TX$(K) + B$: GOTO
 4600
4620  PRINT  CHR$ (1): PRINT D$
;"CLOSE": RETURN
5000  HOME :BL$ = "CONFIGURA E/
S": GOSUB 250
T ";L1: PRINT  TAB( 10)"DRIVE "
;L2
5040  PRINT : PRINT : PRINT  TA
B( 5)"GRAVA EM ": PRINT  TAB( 1
0)"SLOT ";S1: PRINT  TAB( 10)"D
RIVE ";S2
5050  PRINT : PRINT : PRINT "QU
ER ALTERAR? ";
5060  GET R$: IF R$ <  > "S" AN
D R$ <  > "N" THEN 5060
5070  IF R$ = "N" THEN  RETURN
5080  HTAB 1: VTAB 8: CALL  - 9
58: PRINT "CARREGA DE "
5090  VTAB 10: CALL  - 868: INP
UT "SLOT? ";L1
5100  IF L1 < 1 OR L1 > 7 THEN
5090
5110  VTAB 11: CALL  - 868: INP
UT "DRIVE? ";L2
5120  IF L2 <  > 1 AND L2 <  >
2 THEN 5110
5130  VTAB 14: PRINT "GRAVA EM"
5140  VTAB 16: CALL  - 868: INP
UT "SLOT? ";S1
5150  IF S1 < 1 OR S1 > 7 THEN
5140
5160  VTAB 17: CALL  - 868: INP
UT "DRIVE? ";S2
5170  IF S2 <  > 1 AND S2 <  >
2 THEN 5160
5180  GOTO 5020
5200  RETURN : REM  LINHA TEMPO
RARIA
5300  RETURN : REM  LINHA TEMPO
RARIA
5500  HOME :BL$ = "CONFIGURA IM
PRESSORA": GOSUB 250
5510  VTAB 5: PRINT "CONFIGURAC
AO ATUAL:"
5520  PRINT : PRINT : PRINT "LA
RGURA DA LINHA: ";MW;" COL."
5530  PRINT "LARGURA DO TEXTO:
";TW;" COL."
5540  PRINT : PRINT "COMPRIMENT
O DA PAG.: ";PL;" LINHAS"
5550  PRINT "COMPRIMENTO DO TEX
TO: ";TH;" LINHAS"
5560  PRINT : PRINT : PRINT "QU
ER ALTERAR? ";
5570  GET R$: IF R$ <  > "S" AN
D R$ <  > "N" THEN 5570
5580  IF R$ = "N" THEN  RETURN
5590  HTAB 1: VTAB 5: CALL  - 9
58
5600  VTAB 8: INPUT "LARGURA DA
 LINHA? ";MW
5610 MW =  INT (MW): IF MW < 1
THEN 5600
5620  VTAB 9: INPUT "LARGURA DO
 TEXTO? ";TW
5630 TW =  INT (TW): IF TW < 1
OR TW > MW THEN 5620
5640  VTAB 11: INPUT "COMPRIMEN
TO DA PAG.? ";PL
5650 PL =  INT (PL): IF PL < 1
THEN 5640
5660  VTAB 12: INPUT "COMPRIMEN
TO DO TEXTO? ";TH
5670 TH =  INT (TH): IF TH < 1
OR TH > PL THEN 5660
5680 GP =  INT ((MW - TW) / 2):
LF$ = "":LF =  INT ((PL - TH) /
 2)
5690  IF LF = 0 THEN 5500
5700  FOR Z = 1 TO LF:LF$ = LF$
 +  CHR$ (13): NEXT
5710  GOTO 5500
10030  PRINT  CHR$ (4);"PR#1"
10040  PRINT  CHR$ (9);"31N"
10060  LIST 5200 -
10070  PRINT  CHR$ (4);"PR#0"
```

UM SIMULADOR DE VÔO (1)

Nosso programa de simulação é muito semelhante aos utilizados para treinar pilotos em vôo por instrumentos. A seção apresentada neste artigo desenha a cabine de comando.

Em matéria de jogos para computador, variedade é o que não falta: você pode escolher desde a pura fantasia, indo se aventurar num mundo imaginário, até a simulação de situações da vida real. Este último tipo de programa permite que testemos nossa habilidade em enfrentar toda sorte de perigo sem que com isto corramos risco de vida ou de perda material.

Programas de simulação de vôo, especificamente, têm uma aplicação prática muito importante: companhias aéreas e escolas de aviação fazem uso deles no treinamento de seus pilotos. Porém, não falta a tais programas um toque de fantasia: sozinho na cabine de comando, toda a tripulação acometida por uma doença misteriosa, você aterrissa o avião em segurança, usando apenas uma das mãos.

PROGRAMAS DE TREINAMENTO

Para treinar pilotos de verdade, empregam-se simuladores totais ou "fase 3", no jargão de aviação. Estes simuladores, muito sofisticados, permitem que o usuário experimente todas as sensações de um piloto em situação real de vôo. Ele poderá ver o que o piloto vê de sua cabine, terá a impressão de aterrissar, decolar e enfrentar turbulência e ouvirá os sons de um vôo autêntico, inclusive os comandos do controle de tráfego aéreo. Teoricamente, um piloto pode completar seu treinamento em um simulador desse tipo, sem jamais ter saído do chão.

SIMULADORES DE MESA

Os programas de simulação de vôo em microcomputadores são, a exemplo do nosso, bem menos complexos, mas mostram-se também muito úteis para o desenvolvimento dos reflexos do piloto.

Os simuladores de mesa são essenciais para o ensino de vôo por instrumentos, um artifício que permite ao piloto dirigir o aparelho baseado apenas nos instrumentos do painel — o que pode ser necessário no caso de más condições de tempo.

O QUE FAZ NOSSO SIMULADOR

Nosso programa será apresentado em três partes. Ele supõe que o piloto assumiu o comando a 2.000 metros de altitude e a 20.000 metros do centro da pista. Pela janela, pode avistar o horizonte (quando há visibilidade) e um ponto distante, que é a cabeceira da pista. Do mesmo modo que um piloto experiente, você deverá usar o raciocínio para tomar decisões corretas — baseando-se nos instrumentos do painel — e trazer seus passageiros ao solo com segurança.

OS INSTRUMENTOS

O painel tem quatro mostradores. O primeiro informa a velocidade do vento (*airspeed*). Esta varia conforme estejamos mergulhando (a velocidade aumenta), subindo (diminui) ou mudando a potência do motor. Um contador logo abaixo do de velocidade indica a inclinação do vento (*bearing*).

Um segundo mostrador revela o nível do horizonte em relação ao aeroplano. Por meio dele, mesmo sem visibilidade, saberemos onde o horizonte está. O contador logo abaixo aponta a direção da pista de pouso (*runway*).

O terceiro mostrador é um altímetro com dois ponteiros: um para centenas e outro para milhares de metros. O contador abaixo calcula o desvio do aparelho do centro da pista (*drift*). Como ela tem 100 metros de largura, um desvio de + 50 ou -50 ao aterrissarmos será fatal. O último mostrador indica as rotações do motor por minuto (rpm). O contador abaixo fornece a distância do centro da pista.

ATERRISSAGEM

No TRS-Color, a imagem que vemos da cabine torna-se mais clara à medida que nos aproximamos do solo. Ao contrário dos outros micros, onde o piloto precisa se orientar pelos dados do painel, no TRS-Color pode-se observar a pista. Quando assumimos o controle, a pista está ao norte e as condições de tempo são boas. Aterrissar nestas condições é fácil, e o jogo perderia toda a graça se não houvesse outras situações. Para criar dificuldade, podemos mudar a velocidade do vento: uma rajada lateral, por exemplo, dificulta muito a aterrissagem.

O CONTROLE DO AVIÃO

O nível do controle que temos no simulador é o mesmo de que dispõe um piloto em condições reais — embora seja necessário pressionar botões em vez de usar um manche.

- O QUE É SIMULAÇÃO DE VÔO
- PROGRAMAS DE TREINAMENTO
- APRENDENDO A ATERRISSAR
- VÔO POR INSTRUMENTOS
- O PAINEL DO AVIÃO
- COMO CRIAR A SENSAÇÃO DE MOVIMENTO
- UMA ESCOLA DE PILOTAGEM EM CASA

Em um avião, o manche é puxado ou empurrado, movendo os estabilizadores da cauda para cima ou para baixo, conforme estejamos querendo subir ou descer. Usaremos duas teclas para obter tal efeito. A seção do programa que cuida disso será apresentada no terceiro artigo da série.

Para virar o aeroplano, o manche deve ser movido lateralmente, o que aciona os ailerons da asa. Também usaremos teclas para virar o avião.

Dois outros controles nos permitirão acelerar ou retardar o motor, o que é essencial para realizar a aterrissagem de maneira correta ou evitar que o aparelho mergulhe.

VELOCIDADE MÍNIMA

Os aviões não conseguem voar abaixo de uma certa velocidade. Em nosso programa, se a velocidade cair abaixo de 30 metros por segundo, o avião mergulhará, entrando em parafuso. Se estivermos a uma certa altitude, haverá tempo suficiente para evitar um desastre fatal, mas o perigo é iminente.

A DIVISÃO DO PROGRAMA

Dividimos o programa em três partes por ele ser muito longo e complexo.

Na primeira parte, preparamos a tela, desenhando o interior da cabine de comando, com o pára-brisa e os quatro mostradores e contadores. As legendas dos contadores e mostradores estão em inglês, como nos aviões reais.

Os comandos BASIC da listagem já são nossos conhecidos. Os usuários do TRS-Color, contudo, encontrarão um novo comando: **PCOPY**. Este é específico desta linha de micros e, mais tarde, será explicado em detalhes.

Na segunda parte do programa, tomamos os instrumentos do painel sensíveis aos movimentos do avião. Uma seção temporária fará com que o aparelho voe ao acaso, sem piloto, para que possamos

ver os instrumentos funcionando.

A parte final permitirá que você assuma o comando e teste suas habilidades na aterrissagem do avião.

A CABINE DE COMANDO

Para desenhar os instrumentos, digite a primeira parte do programa.

```
  1 POKE 23658,8
 110 GOTO 5000
5000 LET PP=-1: LET RR=-1
5010 LET C=PI/180: LET PY=-2000
: LET PZ=2000: LET AS=150
5110 PLOT 10,175: DRAW 235,0: D
RAW 0,-90: DRAW -235,0: DRAW 0,
90
5120 FOR K=0 TO 3: CIRCLE 35+K*
60,50,20: NEXT K
5130 PRINT AT 12,2;"SPEED   HOR
ZN   ALT    RPM"
5150 PRINT AT 20,0;"BEARING  RU
NWAY  DRIFT  DISTANCE"
5170 PLOT 87,50: DRAW 5,0: DRAW
 3,-3: DRAW 3,3: DRAW 5,0
5180 LET X=35: LET Y=50: GOSUB
7000: LET X=155: GOSUB 7000: LE
T X=215: GOSUB 7000
6900 STOP
7000 FOR K=0 TO 2*PI STEP PI/5:
 PLOT X+17*SIN K,Y+17*COS K: DR
AW 2*SIN K,2*COS K: NEXT K: RET
URN
```

O **POKE** da linha 1 trava o computador em letras maiúsculas. As linhas 5000 e 5010 posicionam o avião no céu: a 2.000 metros de altitude e a 20.000 metros da pista, parado no ar. As linhas que movimentam o aparelho serão apresentadas no próximo artigo.

A linha 5110 desenha a janela do avião, e a 5120, usando um laço **FOR...NEXT**, os mostradores abaixo dela. As linhas 5130 e 5150 imprimem os rótulos para os mostradores. A linha 5170 desenha um diagrama de avião no mostrador de horizonte — o horizonte artificial não será traçado por enquanto. A linha 5180 e a sub-rotina 7000 calculam as posições dos números dos mostradores: velocidade do vento, altitude e rpm. As funções **SIN** e **COS** são usadas conforme explicações dadas no artigo da página 334.

Quando você executar o programa, o interior da cabine aparecerá na tela.

```
10 SCREEN 2,2:COLOR 15,4,2
20 FOR I=0 TO 7*32*8
30 VPOKE BASE(11)+13*32*8+I,8
40 NEXT
110 GOTO 5000
4000 OPEN "GRP:" FOR OUTPUT AS
#1:PRESET(X,Y):PRINT #1,A$
4010 CLOSE #1:RETURN
5000 PP=-1:RR=-1
5010 PI=4*ATN(1):C=PI/180:PY=-2
0000:PZ=2000:VV=150
5110 LINE(10,0)-(245,80),2,B
5120 FOR K=0 TO 3:CIRCLE(35+K*6
0,128),25,2:NEXT
5130 PAINT(255,191),2
5140 X=35:Y=128:GOSUB 7000:X=15
5:GOSUB 7000:X=215:GOSUB 7000
5150 COLOR 1:X=6:Y=88:A$="AIRSP
EED":GOSUB 4000:X=78:A$="HORIZ"
:GOSUB 4000
5160 X=126:A$="ALTITUDE":GOSUB
4000:X=204:A$="RPM":GOSUB 4000
5170 COLOR 15:X=9:Y=160:A$="BEA
RING":GOSUB 4000:X=74:A$="RUNWA
Y":GOSUB 4000
5180 X=138:A$="DRIFT":GOSUB 400
0:X=188:A$="DISTANCE":GOSUB 400
0
5190 DRAW "BM81,126C10R9F5E5R9"
5500 GOTO 5500
7000 FOR K=0 TO 9:LINE(X+21*SIN
(K*PI/5),Y-21*COS(K*PI/5))-(X+1
9*SIN(K*PI/5),Y-19*COS(K*PI/5))
,2:NEXT:RETURN
```

A linha 10 seleciona a tela e suas cores. O **FOR...NEXT** das três linhas seguintes muda a cor de fundo de parte da tela para vermelho médio — **COLOR 8**. Assim, o fundo colorido dos mostradores não sofrerá a interferência de comandos gráficos do BASIC que, em geral, só atuam na cor de frente.

A linha 110 transfere o programa para a linha 5000, que estabelece os valores iniciais de algumas variáveis. Juntamente com a linha 5010, ela posiciona o avião no céu: a 20.000 metros do centro da pista e a 2.000 metros de altitude, parado no ar. As linhas que movimentam o aparelho serão apresentadas no próximo artigo.

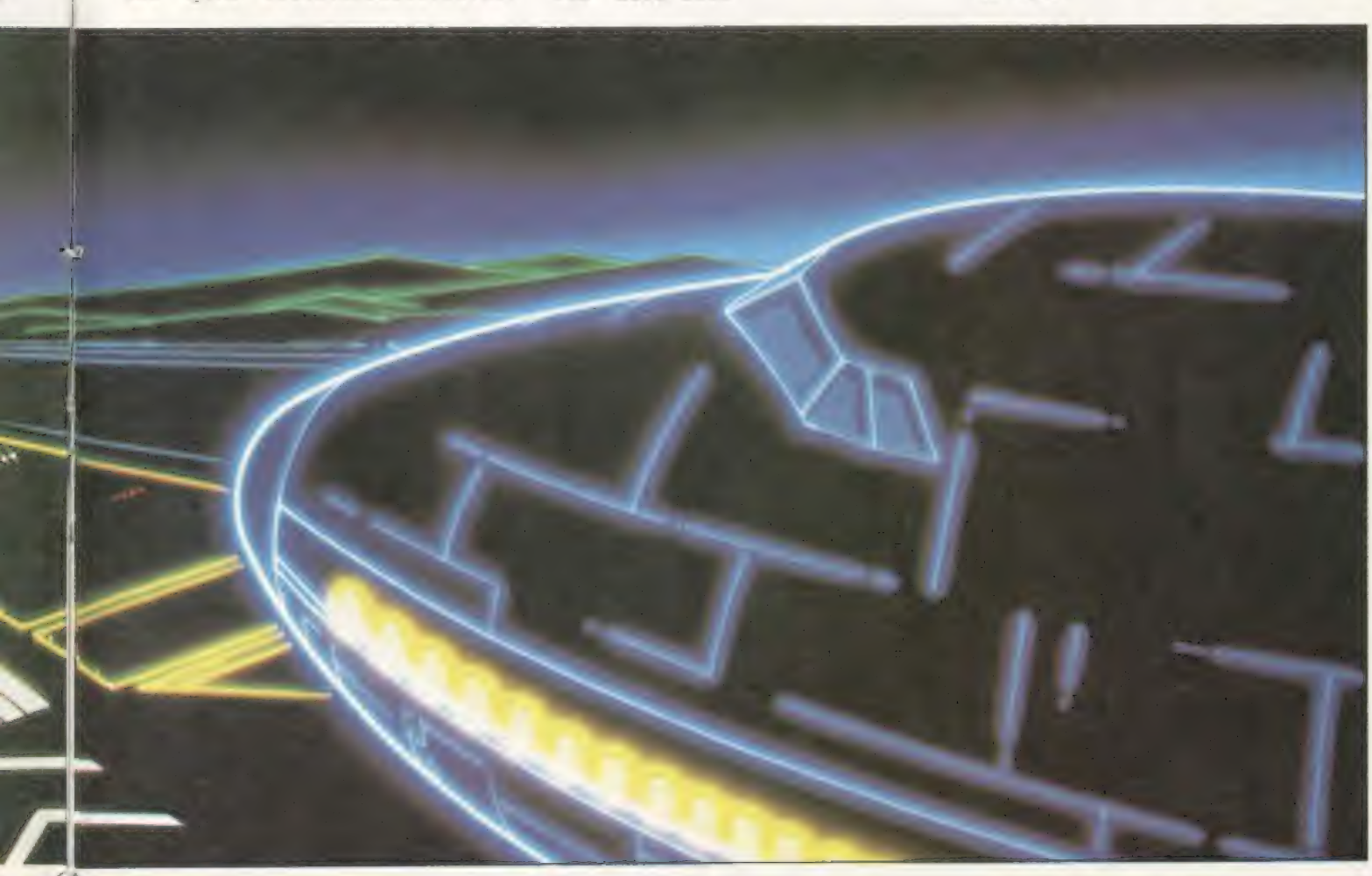
A linha 5110 desenha a janela frontal do avião, e a linha 5120, os círculos dos mostradores. A 5130 colore todo o interior da cabine. A cor usada nessas três linhas deve ser a mesma. As linhas 5140 e 5180 rotulam os mostradores e contadores do painel. Para escrever na tela gráfica é utilizada a sub-rotina da linha 4000. Esta abre um "arquivo" com saída para a tela gráfica — **OPEN "GRP:"** — e imprime o conteúdo da variável alfanumérica A$, a partir das coordenadas X,Y.

A linha 5190 desenha um aeroplano esquemático no mostrador de horizonte. As marcas nos mostradores são desenhadas pela sub-rotina 7000.

Executando o programa agora, o interior da cabine aparecerá na tela.

```
10  HGR2 :E = 35840:T = 16384
20  FOR I = E + 32 * 8 TO E + 3
2 * 8 + 64 * 8 - 1: READ B: POK
E I,B: NEXT
100  DATA  0,0,0,0,0,0,0,0
110  DATA  8,8,8,8,0,0,8,0
120  DATA  18,18,18,0,0,0,0,0
130  DATA  18,18,63,18,63,18,18
,0
140  DATA  8 ,60 ,10 ,28 ,40 ,3
0 ,8 ,0
150  DATA  38,38,16,8,4,50,50,0
160  DATA  12,18,18,12,82,34,92
,0
170  DATA  24,16,8,0,0,0,0,0
180  DATA  32,16,8,8,8,16,32,0
190  DATA  2,4,8,8,8,4,2,0
200  DATA  0,8,42,28,28,42,8,0
210  DATA  0,8,8,62,8,8,0,0
220  DATA   0,0,0,0,0,24,16,8
230  DATA  0,0,0,62,0,0,0,0
240  DATA  0,0,0,0,0,24,24,0
250  DATA  64,32,16,8,4,2,1,0
260  DATA  28,34,38,42,50,34,28
,0
270  DATA  16,24,20,16,16,16,56
,0
280  DATA   28,34,32,24,4,2,62,
0
290  DATA  28,34,32,24,32,34,28
,0
300  DATA  16,24,20,18,62,16,16
,0
310  DATA  62,2,2,30,32,34,28,0
```

```
320  DATA  28,34,2,30,34,34,28,
0
330  DATA   62,32,32,16,8,8,8,0
340  DATA  28,34,34,28,34,34,28
,0
350  DATA  28,34,34,60,32,34,28
,0
360  DATA   0,24,24,0,0,24,24,0
370  DATA  0,24,24,0,0,24,16,8
380  DATA  32,16,8,4,8,16,32,0
390  DATA  0,0,0,62,0,62,0,0
400  DATA  4,8,16,32,16,8,4,0
410  DATA  28,34,32,16,8,8,0,8
420  DATA  28,34,58,42,58,2,60,
0
430  DATA  8,20,34,34,62,34,34,
0
440  DATA  30,34,34,30,34,34,30
,0
450  DATA  28,34,2,2,2,34,28,0
460  DATA  30,34,34,34,34,34,30
,0
470  DATA  62,2,2,62,2,2,62,0
480  DATA  62,2,2,30,2,2,2,0
490  DATA  28,34,2,58,34,34,28,
0
500  DATA  34,34,34,62,34,34,34
,0
510  DATA  24,8,8,8,8,8,28,0
520  DATA  56,16,16,16,18,18,28
,0
530  DATA  34,18,10,6,10,18,34,
0
540  DATA  2,2,2,2,2,2,62,0
550  DATA  65,99,85,73,65,65,65
,0
560  DATA  34,38,42,42,50,34,34
,0
570  DATA  28,34,34,34,34,34,28
,0
580  DATA   30,34,34,30,2,2,2,0
590  DATA  28,34,34,34,42,50,60
,64
600  DATA  30,34,34,30,18,34,34
,0
610  DATA  60,2,2,28,32,32,30,0
620  DATA  62,8,8,8,8,8,8,0
630  DATA  34,34,34,34,34,34,28
,0
640  DATA  34,34,34,34,34,20,8,
0
650  DATA  65,65,65,73,85,99,65
,0
660  DATA  34,34,20,8,20,34,34,
0
670  DATA   34,34,34,20,8,8,8,0
680  DATA  62,32,16,8,4,2,62,0
690  DATA  60,4,4,4,4,4,60,0
700  DATA  1,2,4,8,16,32,64,0
710  DATA  30,16,16,16,16,16,30
,0
720  DATA  8,20,34,0,0,0,0,0
730  DATA  0,0,0,0,0,0,62,0
740  GOTO 5000
4000 N = L * 40 + C
4010  FOR J = 1 TO  LEN (A$)
4020 B$ =  MID$ (A$,J,1)
4030 B =  ASC (B$)
4040 L =  INT (N / 40):C = N -
40 * L
4050  FOR I = 0 TO 7
4060  POKE T + (L - 8 * (L > 7)
 - 8 * (L > 15)) * 128 + 40 * (
L > 7) + 40 * (L > 15) + C + 10
24 * I, PEEK (E + B * 8 + I)
4070  NEXT I:N = N + 1: NEXT J
4080  RETURN
5000 PP =  - 1:RR =  - 1
5010 PI = 4 *  ATN (1):C = PI /
 180:C1 = PI / 10:PY =  - 20000
:PZ = 2000:AS = 150
5110  HGR2 : HCOLOR= 6: HPLOT 1
0,0 TO 276,0 TO 276,80 TO 10,80
 TO 10,0
5120  HCOLOR= 3: FOR K = 0 TO 3
: HPLOT 71 + K * 68,134: FOR I
= 0 TO 10 STEP PI / 10: HPLOT
TO 42 + K * 68 + 30 *  COS (C1
+ I * PI / 5),134 + 26 *  SIN (
C1 + I * PI / 5): NEXT I,K
5130 C = 2:L = 12:A$ = "AIRSPEE
D": GOSUB 4000:C = 12:A$ = "HOR
IZON": GOSUB 4000
5140 C = 22:A$ = "ALTITUDE": GO
SUB 4000:C = 33:A$ = "RPM": GOS
UB 4000
```



```
5150 C = 2:L = 21:A$ = "BEARING
": GOSUB 4000:C = 13:A$ = "RUNW
AY": GOSUB 4000
5160 C = 23:A$ = "DRIFT": GOSUB
 4000:C = 31:A$ = "DISTANCE": G
OSUB 4000
5170  HCOLOR= 5: HPLOT 85,134 T
O 104,134 TO 109,144 TO 114,134
 TO 134,134
5180  HCOLOR= 3: FOR K = 0 TO 3
: IF K = 1 THEN  NEXT K
5190  FOR I = 0 TO 9: HPLOT 42
+ K * 68 + 28 *  COS (C1 + I *
PI / 5),134 + 24 *  SIN (C1 + I
 * PI / 5) TO 42 + K * 68 + 26
*  COS (C1 + I * PI / 5),134 +
22 *  SIN (C1 + I * PI / 5): NE
XT I,K
```

Os usuários do TK-2000, além de não copiarem as linhas de 10 a 4080, devem fazer as seguintes modificações no programa.

```
740  GOTO 5000
4000  HTAB C + 1: VTAB L + 1: P
RINT A$
4010  RETURN
5100 MP
5120  HCOLOR= 3: FOR K = 0 TO 3
:XX = 71 + K * 68:YY = 134: FOR
 I = 0 TO 10 STEP PI / 10:X = 4
2 + K * 68 + 30 *  COS (C1 + I
* PI / 5):Y = 134 + 26 *  SIN (
C1 + I * PI / 5): HPLOT XX,YY T
O X,Y:XX = X:YY = Y: NEXT I,K
```

As linhas iniciais criam um banco de blocos no topo da memória, para que o Apple possa escrever na tela. Essas linhas já foram apresentadas em artigo anterior, quando explicamos o uso de blocos gráficos: pode ser, portanto, que você não precise digitá-las novamente. O TK-2000 pode escrever na tela gráfica sem esse artifício; assim, seus usuários devem começar a digitação a partir da linha 5000.

Como vamos utilizar a página 2, armazenamos o banco de blocos em outra área da memória, devidamente reservada por meio de **HIMEM:**.

A sub-rotina que começa na linha 4000, responsável pela impressão de palavras na tela gráfica, é essencialmente a mesma publicada em artigo anterior, só que com outro número de linhas. Os usuários do TK-2000 dispõem de uma versão bem mais simples dessa sub-rotina. Na realidade, ela poderia ser dispensada neste micro; resolvemos introduzi-la aqui para que os programas do Apple e do TK-2000 não ficassem muito diferentes.

As linhas 5000 e 5010 posicionam o avião no céu: a 20.000 metros do centro da pista e a 2.000 metros de altitude, parado no ar. As linhas que movimentam o aparelho serão apresentadas no próximo artigo.

A linha 5110 desenha o pára-brisa do avião. No TK-2000 há um comando **MP** para ativar a segunda página de vídeo; no Apple, isso é feito por **HGR2**. Note que o programa do TK-2000 deve conter **HGR2** também, para permitir o uso das linhas inferiores da tela.

A linha 5120, um pouco diferente para os dois micros, desenha os círculos dos mostradores. As linhas 5130 a 5160 rotulam os mostradores e contadores do painel em inglês, como nos aviões reais. A 5170 traça o diagrama de um aeroplano no mostrador de horizonte. As marcas nos mostradores são feitas pelas linhas 5180 e 5190.

Ao ser executado, o programa desenhará o interior da cabine de nosso aparelho imaginário.

```
10 PCLEAR 8:PMODE 4,1
20 DIM LE$(26)
30 FOR K=0 TO 26:READ LE$(K):NE
XT
40 FOR K=0 TO 9:READ NU$(K):NEX
T
50 DATA BR2,ND4R3D2NL3ND2BE2,ND
4R3DGNL2FDNL3BU4BR2,NR3D4R3BU4B
R2,ND4R2FD2GL2BE4BR,NR3D2NR2D2R
3BU4BR2
60 DATA NR3D2NR2D2BE4BR,NR3D4R3
U2LBE2BR,D4BR3U2NL3U2BR2,ND4BR2
,BD4REU3L2R3BR2,D2ND2NF2E2BR2
70 DATA D4R3BU4BR2,ND4FREND4BR2
,ND4F3DU4BR2,NR3D4R3U4BR2,ND4R3
D2NL3BE2,NR3D4R3NHU4BR2
80 DATA ND4R3D2L2F2BU4BR2,BD4R3
U2L3U2R3BR2,RND4RBR2,D4R2U4BR2,
D3FEU3BR2,D4EFU4BR2
90 DATA DF2DBL2UE2UBR2,DFND2EUB
R2,R3G3DR3BU4BR2
100 DATA NR2D4R2U4BR2,BDEND4BR2
,R2D2L2D2R2BU4BR2,NR2BD2NR2BD2R
2U4BR2,D2R2D2U4BR2,NR2D2R2D2L2B
E4,D4R2U2L2BE2BR2,R2ND4BR2,NR2D
4R2U2NL2U2BR2,NR2D2R2D2U4BR2
110 GOTO 5000
4000 FOR K=1 TO LEN(A$)
4010 B$=MID$(A$,K,1)
4020 IF B$>="0" AND B$<="9" THE
N DRAW NU$(VAL(B$)):GOTO 4050
4030 IF B$=" " THEN N=0 ELSE N=
ASC(B$)-64
4040 DRAW LE$(N)
4050 NEXT :RETURN
5000 PP=-1:RR=-1
5010 PI=4*ATN(1):C=PI/180:PY=-2
0000:PZ=2000:VV=150
5110 PCLS:LINE(10,0)-(245,80),P
SET,B
5120 FOR K=0 TO 3:CIRCLE(35+K*6
0,120),25,5:NEXT
5130 DRAW "BM18,88S4":A$="AIRSP
EED":GOSUB 4000:DRAW "BM80,88":
A$="HORIZON":GOSUB 4000
5140 DRAW "BM140,88":A$="ALTITU
DE":GOSUB 4000:DRAW"BM208,88":A
$="RPM":GOSUB 4000
5150 DRAW "BM18,160":A$="BEARIN
G":GOSUB 4000:DRAW"BM82,152":A$
="RUNWAY":GOSUB 4000:DRAW"BM80,
160":A$="BEARING":GOSUB 4000
5160 DRAW"BM144,160":A$="DRIFT"
:GOSUB 4000:DRAW"BM200,160":A$=
"DISTANCE":GOSUB 4000
5170 DRAW"BM81,118R9F5E5R9"
5180 X=35:Y=120:GOSUB 7000:X=15
5:GOSUB 7000:X=215:GOSUB 7000
5190 PCOPY 3 TO 5:PCOPY 3 TO 7:
PCOPY 4 TO 6:PCOPY 4 TO 8:SCREE
N 1,1
5500 GOTO 5500
7000 FOR K=0 TO 9:LINE(X+24*SIN
(K*PI/5),Y-24*COS(K*PI/5))-(X+2
1*SIN(K*PI/5),Y-21*COS(K*PI/5))
,PSET:NEXT:RETURN
```

A primeira parte do simulador inclui um comando ainda não utilizado em INPUT: **PCOPY**, que assegura a movimentação suave da imagem na tela.

As linhas 20 a 110 preparam as matrizes que não contêm os blocos gráficos necessários ao desenho da cabine. A sub-rotina das linhas 4000 e 4050 imprime os gráficos na tela.

As linhas 5000 e 5010 posicionam o aparelho no céu: a 20.000 metros do centro da pista e a 2.000 metros de altitude, parado no ar. As linhas que movimentam o aparelho serão apresentadas no próximo artigo.

A linha 5110 desenha o pára-brisa, 5120 desenha os círculos dos mostradores, e as linhas 5130 a 5160 rotulam os mesmos — em inglês, como nos aviões reais. A linha 5170 coloca o diagrama de um avião no mostrador de horizonte. As marcas nos mostradores são desenhadas pela linha 5180 e pela sub-rotina da linha 7000.

Os comandos **PCOPY** na linha 5190 desenham gráficos em páginas ainda invisíveis e depois copiam-nos na tela. Assim, auxiliam a preservar os desenhos do fundo, que são atualizados antes de serem transferidos para a tela, de modo que o lapso de tempo entre um quadro e outro seja mínimo. Isso significa que, quando o aeroplano está em movimento, os mostradores são alterados de acordo, simultaneamente. Sem **PCOPY**, apenas um mostrador poderia ser atualizado de cada vez.

Ao executar esta seção do programa, o interior da cabine aparecerá desenhado na tela.

No próximo artigo, apresentaremos as linhas que fazem o avião voar, embora ainda sem controle. Ele vagará pelos céus, mergulhando e subindo ao acaso. Você poderá controlá-lo quando concluir o programa, no terceiro e último artigo. Finalmente, a vida dos passageiros estará em suas mãos.

AMPLIE O BASIC DO TRS-COLOR

- PLANEJAMENTO DAS NOVAS INSTRUÇÕES
- O QUE É UM **STUB**
- NOVOS COMANDOS NO MICRO
- RECUPERAÇÃO DE PROGRAMAS

O BASIC do TRS-Color não é intocável. Pode-se modificá-lo para incluir novos comandos, o que é útil sobretudo quando o micro executa tarefas que empregam certas rotinas repetidas vezes.

O BASIC é apenas um programa em linguagem de máquina funcionando em seu microcomputador. Ainda que as instruções executadas estejam codificadas na memória ROM, no TRS-Color é possível fazer algumas modificações para criar novos comandos.

Esta possibilidade revela-se bastante útil nos casos em que usamos o computador para realizar uma tarefa específica que emprega determinadas rotinas repetidas vezes. Neste artigo, criaremos duas novas instruções destinadas ao micro TRS-Color.

A rotina INVERT (também chamada de INVERSE em outros microcomputadores) liga todos os pixels da tela que estavam desligados e desliga todos os que estavam ligados. Assim, tudo que estava representado na cor de um grupo assume a cor correspondente ao outro grupo.

A rotina que se segue acrescenta dois comandos ao BASIC do TRS-Color: **OLD**, que permite a recuperação de um programa apagado acidentalmente com **NEW**, e **INVERT**, que troca o grupo de quatro cores que estiver sendo utilizado na tela gráfica.

```
10   ORG 31000
20   SETUP LDX #298
30   LDU #308
40   STONE LDA ,X+
50   STA ,U+
60   CMPX #308
70   BLO STONE
80   LDA #2
90   STA 298
100  LDX #NEWRDS
110  STX 299
120  LDX #NEWDSP
130  STX 301
140  LDX #NEWUSR
150  STX 176
160  LDU #$8B8D
170  LDA #10
180  STTWO STU ,X++
190  DECA
200  BNE STTWO
210  RTS
220  NEWRDS FCB 79,76,196
230  FCB 73,78,86,69,82,212
240  NEWDSP CMPA #$CE
250  BLO NDONE
260  CMPA #$D0
270  BHS NDONE
280  SUBA #$CE
290  LDX #NEWTBL
300  JMP $84ED
310  NDONE JMP $89B4
320  OLD LDU 25
330  PSHS U
340  LEAX 4,U
350  OLDONE LDA ,X+
360  BNE OLDONE
370  TFR X,D
380  SUBD ,S++
390  TSTA
400  BEQ OLDTWO
410  JMP $8B8D
420  OLDTWO STX ,U
430  OLDTHR TFR X,U
440  LDX ,U
450  BNE OLDTHR
460  LEAU 2,U
470  STU 27
480  STU 29
490  STU 31
500  RTS
510  INVERT LDA 65314
520  EORA #8
530  STA 65314
540  RTS
550  NEWTBL FDB $7964
560  FDB $7989
570  NEWUSR EQU *
```

O QUE É UM STUB

Para acrescentar comandos ao programa BASIC, é necessário que reservemos as palavras correspondentes. Temos, assim, que ampliar a lista de palavras reservadas, bem como dirigir o BASIC aos programas em código que executam as novas funções.

A posição das rotinas que executam os comandos BASIC é apontada por determinados vetores — ou apontadores — que ficam em porções da memória denominadas *stubs*. Existem normalmente dois stubs, cada um ocupando dez bytes da memória RAM. O segundo deles, porém, é só um sinalizador de final de tabela, não tendo outra função.

Quando acrescentamos instruções devemos criar também um stub que aponte para as novas rotinas. Mas, antes disso, é preciso deslocar o stub que indica o fim da tabela para uma posição mais alta na memória, criando o espaço necessário.

O endereço inicial do segundo stub é 298, e as primeiras seis instruções do programa aqui apresentado transferem-no para 308. O registro X é usado como apontador para o stub original, enquanto o registro U aponta para sua nova posição. Os operandos **,X+** e **,U+** incrementam esses apontadores a cada volta do laço **STONE**.

Este programa não pode ser usado em sistemas com disquete, pois os comandos de disco usam o mesmo stub.

CRIAÇÃO DE UM NOVO STUB

Uma vez garantido o espaço necessário na memória de acesso aleatório e terminado o laço **STONE**, passamos à criação do novo stub.

O primeiro byte desse novo stub corresponde ao número de instruções apontadas por ele. Portanto, **LDA #2** e **STA 298** colocam o número 2 no primeiro byte.

Os dois próximos bytes trazem o endereço da tabela de nomes de comandos. Estes nomes aparecem imediatamente após o rótulo **NEWRDS**, com suas letras codificadas uma a uma em ASCII. Só não ocorre o mesmo com a última letra da cada palavra, que se distingue por ter seu bit mais significativo ativado. Assim, D — última letra do comando **OLD** — é representada pelo seu código ASCII, 68, mais 10000000 em binário ou 128 em decimal (68 + 128 = 196). Da mesma maneira, o T de **INVERT** será 212 e não 84.

Finalmente, quarto o e o quinto bytes trazem o endereço inicial das rotinas que executam os comandos. Esse endereço é encontrado nos bytes após o rótulo **NEWTBL**.

MICRO DICAS

ADICIONE SEUS PRÓPRIOS COMANDOS

Depois de entender como funcionam os *stubs* no reconhecimento dos comandos BASIC, o programador pode criar seus próprios comandos. Quem já tem uma certa experiência deve começar pelos comandos sem operandos, como os deste artigo. Qualquer rotina pode se transformar num comando desse tipo. Basta o BASIC encontrar o novo comando em uma linha de programa, e o controle é desviado para a rotina em código cujo endereço está no *stub*. Após sua execução, o controle retorna ao interpretador.

Comandos que operam sobre números ou variáveis exigem muito do programador. Se os números não forem inteiros, ele terá de conhecer os segredos da aritmética de ponto flutuante e dos números decimais codificados em binário. Além de um bom conhecimento de linguagem de máquina, é preciso um profundo domínio da arquitetura do TRS-Color; ao mesmo tempo, deve-se saber onde ficam as variáveis do sistema, como são armazenadas as variáveis e as linhas BASIC, e onde podem ser colocados temporariamente os valores dos operandos.

VETORES USR

O vetor que fica nos endereços 176 e 177 aponta para as posições que contêm o endereço de rotinas **USR**. Estas posições ficam geralmente logo acima do segundo stub, em uma tabela que começa em 308. O segundo stub, contudo, foi empurrado para dentro dessa área para criar espaço para o novo stub, de forma que a área **USR** também precisa ser relocada.

LDX #NEWUSR e **STX 176** colocam o endereço do rótulo **NEWUSR** nas posições 176 e 177. **EQU ***, que fica logo após **NEWUSR**, no final do programa, reserva os próximos bytes para a tabela **USR**.

OS CÓDIGOS DOS NOVOS COMANDOS

O código de maior valor usado pelo BASIC é CD. Assim, os códigos dos dois novos comandos serão CE e CF.

CMPA #$CE e **BLO NDONE** verificam se o código da instrução está abaixo do valor dos novos códigos. Se isso ocorrer e o primeiro stub não tiver reconhecido o comando, houve um erro

de sintaxe. O microprocessador é, então, enviado ao rótulo **NDONE** que, por sua vez, provoca um salto para o endereço 89B4 — uma sub-rotina da ROM que emite uma mensagem de erro.

CMPA #$0 e **BHS NDONE** fazem o mesmo se o código for maior que CF, ou seja, se ele estiver acima da faixa em que ficam os códigos das novas instruções.

Se o código passar por esses dois testes, fica comprovado que ele corresponde a um dos novos comandos. O programa, então, segue em frente. CE é subtraído do valor do código, que está em A. O resultado permanece em A.

O endereço inicial da tabela de endereços dos novos comandos é colocado em X. Assim, quando o processador vai para 84ED — uma rotina que cuida do reconhecimento dos vários comandos BASIC —, leva consigo os valores contidos em X e em A.

A ROTINA DO COMANDO OLD

Quando usamos **NEW** para apagar um programa, os valores de diversas variáveis do sistema modificam-se. Se restaurarmos os valores dessas variáveis antes de digitarmos outro programa ou de usarmos um comando **PCLEAR**, poderemos recuperar o programa antigo. O comando **NEW** modifica os dois primeiros bytes da primeira linha do programa que se quer recuperar. Esses bytes indicam o endereço inicial da linha seguinte. Outras variáveis do sistema que apontam o final da área do programa BASIC também são modificadas. O que o comando **OLD** faz é restaurar o valor original de todos esses apontadores.

A rotina que cuida disso começa com a instrução **LDU 25**, que coloca o conteúdo das posições 25 e 26 no registro U. Elas contêm o endereço inicial do programa BASIC. **PSHS U** coloca este valor na pilha, para que possamos recuperá-lo mais tarde.

A instrução **LEAX 4,U** cuida do endereço do byte 4, contado a partir do início do BASIC. Os dois primeiros bytes — 0 e 1 — continham o endereço da próxima linha. Os dois seguintes — 2 e 3 —, o número da linha. Assim, U contém agora o primeiro byte da linha propriamente dita.

LDA ,X+ coloca este byte no acumulador e incrementa X. **BNE OLDONE** faz o processador repetir essa instrução até encontrar um zero, que sinaliza o final da linha BASIC.

Quando o programa encontra um zero e sai do laço, X já foi incrementado e aponta para o endereço inicial da próxima linha BASIC.

VERIFICAÇÃO DE ERROS

Se o comando **OLD** for acionado sem que haja um programa BASIC aproveitável na memória, devemos impedir que o lixo que porventura esteja na área BASIC seja recuperado. Precisamos, assim, de um teste que verifique se a primeira linha ainda faz sentido.

Para fazer isso, o conteúdo do apontador X é colocado em D e dele subtraímos o último item da pilha — o endereço inicial do BASIC — com **SUBD ,S++**. O resultado indica o comprimento da primeira linha, que permanece armazenada em D. O apontador da pilha é decrementado, o que retira o endereço inicial do BASIC da pilha.

TSTA testa o acumulador A, isto é, verifica o conteúdo do byte mais significativo de D (os registros A e B juntos constituem o registro D) e estabelece os sinalizadores de acordo. **BEQ OLDTWO** salta as instruções seguintes, se o sinalizador zero foi ativado. Se o conteúdo de A não for zero, o registro D continha um valor maior que 255 — o maior número de bytes que uma linha BASIC pode conter. Nesse caso, o processador é enviado para a rotina da memória ROM, em 8B8D, que emite uma mensagem de erro.

RESTAURE OS APONTADORES

O registro U aponta para o primeiro byte da primeira linha BASIC. Assim, para restaurar o apontador do início da linha seguinte, colocamos o valor de X no endereço apontado por U, usando **STX ,U**.

O passo seguinte é descobrir onde fica o final da área do programa BASIC. O laço **OLDTHR** trata disso.

Como os dois primeiros bytes de qualquer linha apontam para o início da seguinte, é fácil saltar de linha em linha, transferindo o conteúdo de X para U e colocando em X o valor contido nas posições apontadas por U. X contém o endereço da próxima linha, que é colcado em U. U aponta, então, para os dois bytes que contêm o endereço da linha que fica ainda mais adiante, este valor é colocado em X.

O mesmo laço é executado repetidas vezes, até que se encontre uma linha cujos dois primeiros bytes sejam 00 00. Este é o final do programa BASIC. Quando ele é encontrado, o processador sai do laço.

O endereço que fica logo acima desses dois últimos bytes é colocado em U, que passa a apontar para o início da área das variáveis.

STU 27, STU 29, e **STU 31** copiam esse endereço nas variáveis do sistema que ficam em 28 e 29, apontando para o início da área de variáveis; em 29 e 30, apontando para o início da tabela de matrizes; e em 31 e 32, apontando para o topo da RAM usada.

RTS retorna ao BASIC.

A ROTINA INVERT

A posição de memória FF22 fica na memória de entrada e saída — I/O — de periféricos e controla a saída de dados gráficos para a tela. O bit 3 dessa posição determina o grupo de cores que está sendo utilizado.

LDA $FF22 coloca o conteúdo daquele endereço no acumulador e o bit 3 é alterado pela operação lógica "ou exclusivo", entre A e 8 (00001000, em binário). O resultado é recolocado em FF22, e **RTS** volta ao BASIC.

Essa rotina simplesmente alterna dois grupos de cores, fazendo com que tudo o que está representado na cor de um grupo assuma a cor correspondente do outro grupo de cores.

O programa não é recolocável.

Só há uma forma de o micro aceitar instruções adicionais?

Não. Embora os diversos modelos de micro tenham um BASIC semelhante, o programa que o entende varia muito de um para outro: alguns não têm *stubs*, e novas instruções devem ser adicionadas sem estes.

Podemos chamar de "sistema" o programa existente na ROM que entende o BASIC. Ele deve ser muito bem organizado para poder associar a cada comando a rotina em código certa (muitas vezes os comandos agem sobre operandos, o que complica ainda mais a tarefa do sistema). Assim, ele mantém diversas tabelas com endereços e valores que o orientam na interpretação do BASIC. Os *stubs* nada mais são que tabelas desse tipo. Mas existem outras. A principal maneira de se adicionar novos comandos ao BASIC de uma máquina consiste em alterar determinados valores nessas tabelas de controle.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

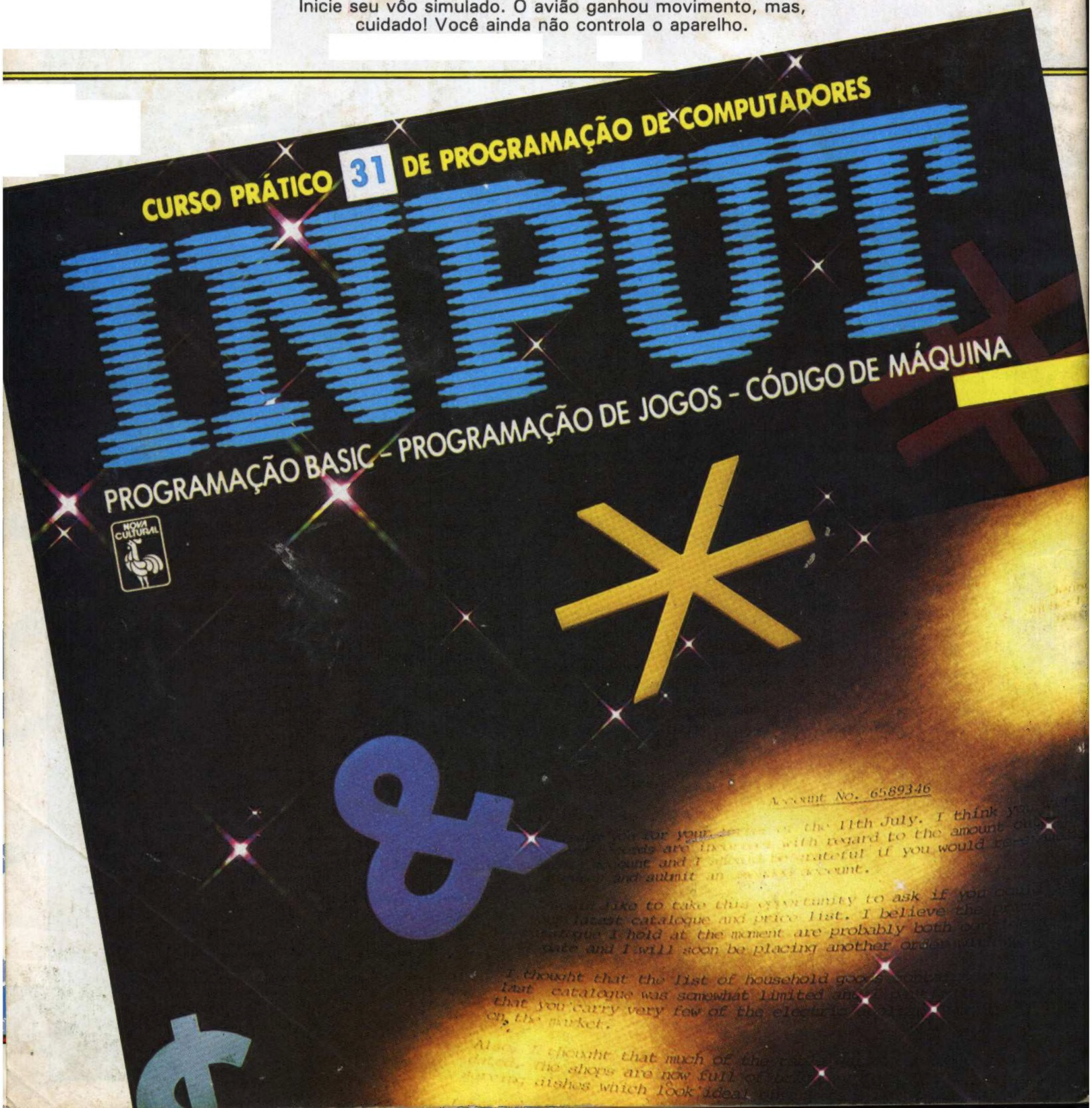
NO PRÓXIMO NÚMERO

APLICAÇÕES

Você já domina as funções básicas de edição. Conheça agora as rotinas e símbolos mais úteis à preparação de um texto.

PROGRAMAÇÃO BASIC

O interpretador BASIC de seu micro conta com várias funções matemáticas padronizadas. Veja como definir algumas outras.

PROGRAMAÇÃO DE JOGOS

Inicie seu vôo simulado. O avião ganhou movimento, mas, cuidado! Você ainda não controla o aparelho.